



米思齐Arduino Easy学习套件

睿龙创客工场出品



米思齐Arduino Easy 学习套件



目录

Arduino Easy

- [套件简介](#)
- [套件展示](#)
- [硬件功能介绍](#)
- [米思齐](#)
- [Arduino IDE](#)

基础应用

- [流水灯](#)
- [流光沙漏](#)
- [计时红绿灯](#)
- [可调台灯](#)
- [人体感应灯](#)
- [光线检测器](#)
- [可调RGB彩灯](#)
- [触摸风扇](#)
- [温度检测器](#)
- [温湿度检测器](#)
- [磁性探测器](#)
- [声音检测器](#)
- [入侵报警](#)

- [模拟倒车雷达](#)
 - [红外遥控灯](#)
 - [红外遥控调速风扇](#)
 - [中英文显示](#)
 - [奥运五环](#)
 - [真心话大冒险](#)
- ## Arduino片上应用
- [模拟汽车双闪灯](#)
 - [按键倒计时器](#)
 - [音乐门铃](#)
 - [定时器](#)
 - [多线程](#)
 - [双人抢答器](#)
 - [EEPROM](#)
 - [状态转换灯](#)
 - [串口交互灯](#)
- ## 综合应用
- [幸运大转盘](#)
 - [智能楼道感应灯](#)
 - [声光控路灯](#)

- [车位检测](#)
- [特种车辆报警](#)
- [噪声检测仪](#)
- [穿衣提醒](#)
- [电子蜡烛](#)
- [流光溢彩音乐盒](#)
- [防扒窗报警](#)
- [车辆防盗报警](#)
- [语音温度计](#)
- [语音光线强度计](#)
- [语音温湿度计](#)
- [语音测距仪](#)
- [简易交通灯](#)
- [秒表](#)
- [触摸倒计时器](#)
- [智能坐姿提醒](#)
- [红外遥控感应风扇](#)
- [红外遥控调光灯](#)
- [红外遥控钢琴](#)



米思齐Arduino Easy学习套件，是适用学校、社团、个人学习入门Arduino学习套件，套件收录了28个常用传感器，采用睿龙创客工场自行设计的UNO R3兼容板为主板，以UNO R3防呆IO扩展板，扩展接插传感器模块，杜绝差错，配套三大篇幅的课程内容，从传感器基础原理应用，到Arduino片上资源以及编程技巧，再至综合案例实战，全方面入门精通学习Arduino硬件知识，软件上配套Mixly结合代码方式进行学习深入。

软件支持：Mixly\Arduino IDE等，附带标准库
注：本套件以官方版本Mixly1.7讲解说明。





- 1pcs x UNO R3 For Maker (兼容版)
- 1pcs x UNO R3 Easy IO Shield Board (UNO R3防呆IO扩展板)
- 1pcs x 闪灯模块 (红)
- 1pcs x 闪灯模块 (绿)
- 1pcs x 闪灯模块 (黄)
- 1pcs x LED模块 (绿)
- 1pcs x LED模块 (红)
- 1pcs x 按钮传感器 (白)
- 1pcs x 按钮传感器 (绿)
- 1pcs x 数字智能热释电红外传感器
- 1pcs x 声音传感器 (MV358I)
- 1pcs x 光敏传感器(PT550)
- 1pcs x 温度传感器(LM35)
- 1pcs x 温度传感器 (DS18B20) 转接板
- 1pcs x 温湿度传感器(DHT11)
- 1pcs x 磁敏传感器 (GPS-14A)
- 1pcs x LED 灯串模块 (10灯暖白)
- 1pcs x 触摸传感器 (点动)



- 1pcs x 马达风扇模块 (C300)
- 1pcs x 蜂鸣器模块
- 1pcs x 喇叭扬声器模块
- 1pcs x HC-SR04+超声波传感器 3.3/5V宽电压 定制版
- 1pcs x I2C LCD1602字符液晶显示器
- 1pcs x 红外接收传感器
- 1pcs x 红外软硅胶遥控器 (Ruilingmaker)
- 1pcs x 震动传感器]
- 1pcs x RGB彩灯模块(4*WS2812)
- 1pcs x I2C 0.56寸数码管模块(TM1650)
- 1pcs x 旋钮传感器
- 1pcs x 温度传感器-DS18B20 不锈钢封装 防水型
- 1pcs x I2C 0.96寸OLED模块(SSD1306)
- 1pcs x 语音模块 (68段日常用语)
- 1pcs x USB线-高品质 Micro USB线 (黑色) 2A 新款订货
- 1pcs x 传感器线-双头XH2.54 3Pin 15cm
- 1pcs x 传感器线-双头XH2.54 4Pin 15cm
- 1pcs x G360元件盒 36*22*5



功能介绍

序号	名称	备注
1	闪灯模块	亮（高电平），灭（低电平）
2	LED模块	亮（高电平），灭（低电平）
3	按钮传感器	按下（低电平），抬起（高电平）
4	热释电传感器	默认为（低电平），检测到人为（高电平）
5	声音传感器	声音↑，数值↑
6	光线传感器	光线强度↑，数值↑
7	温度传感器LM35	模拟传感器，温度↑，数值↑
8	温度传感器	数字传感器
9	温湿度传感器	单总线采集
10	磁敏传感器	默认为高电平（1），检测到磁性为低电平（0）
11	LED灯串模块	亮起（高电平），熄灭（低电平）
12	LCD1602显示屏	显示（高电平），熄灭（低电平）



功能介绍

序号	名称	备注
13	红外接收传感器	默认为低电平，接收到红外信号为高电平
14	触摸传感器	默认为低电平，触摸为高电平
15	蜂鸣器模块	蜂鸣（高电平），停止（低电平），默认为低。
16	震动传感器	默认为低，检测到震动，为高电平。
17	RGB彩灯模块	亮起（高电平），熄灭（低电平）
18	语音模块	68段常用语音，
19	旋钮传感器	顺时针，数值增大；逆时针，数值减小
20	喇叭扬声器模块	播放（高电平），否则为（低电平）
21	超声波传感器	IO触发测距，给至少10us高电平信号，发送信号并检测，有信号返回，通过IO输出一高电平，高电平持续的时间就是超声波从发射到返回的时间。测试距离=(高电平时间*声速(340M/S))/2;
22	马达风扇模块	风扇转动（高电平），否则为低电平
23	OLED模块	显示（高电平），熄灭（低电平）
24	0.56寸数码管模块	显示（高电平），熄灭（低电平）



软件下载

The screenshot shows a Wikipedia-style page for Mixly. The top navigation bar includes 'Mixly Wiki' and 'latest'. The main content area has a breadcrumb trail: 'Docs > 软件使用基础 > 软件安装与更新'. On the right, there's a 'View page source' link. The main title is '软件安装与更新'. Below it, a note states: '目前最新版本为1.0.0，支持Windows 7/8/10、MacOS、Linux(非arm框架)。' The left sidebar contains a navigation menu with sections like '软件使用基础' (selected), 'Mixly 简介', '软件安装与更新' (selected), 'Windows版本安装' (selected), 'Mac版本安装', '界面介绍', '支持板卡', '范例', '自定义公司库', '第三方库开发', 'FAQ', 'Arduino AVR编程', and 'Arduino ESP8266编程'.

软件安装与更新

目前最新版本为1.0.0，支持Windows 7/8/10、MacOS、Linux(非arm框架)。

下载软件

- 【Window系统 地址1】 Mixly For Win7or10
- 【Window系统 地址2】 Mixly For Win7or10
- 【Mac系统】 Mixly For Mac

Windows版本安装

安装软件

下载Mixly_WIN.7z压缩包，右键解压到本地磁盘。

注解

- 建议解压到硬盘根目录，路径不能包含中文及特殊字符(如: _ ()等)。
- 建议安装路径如E:Mixly

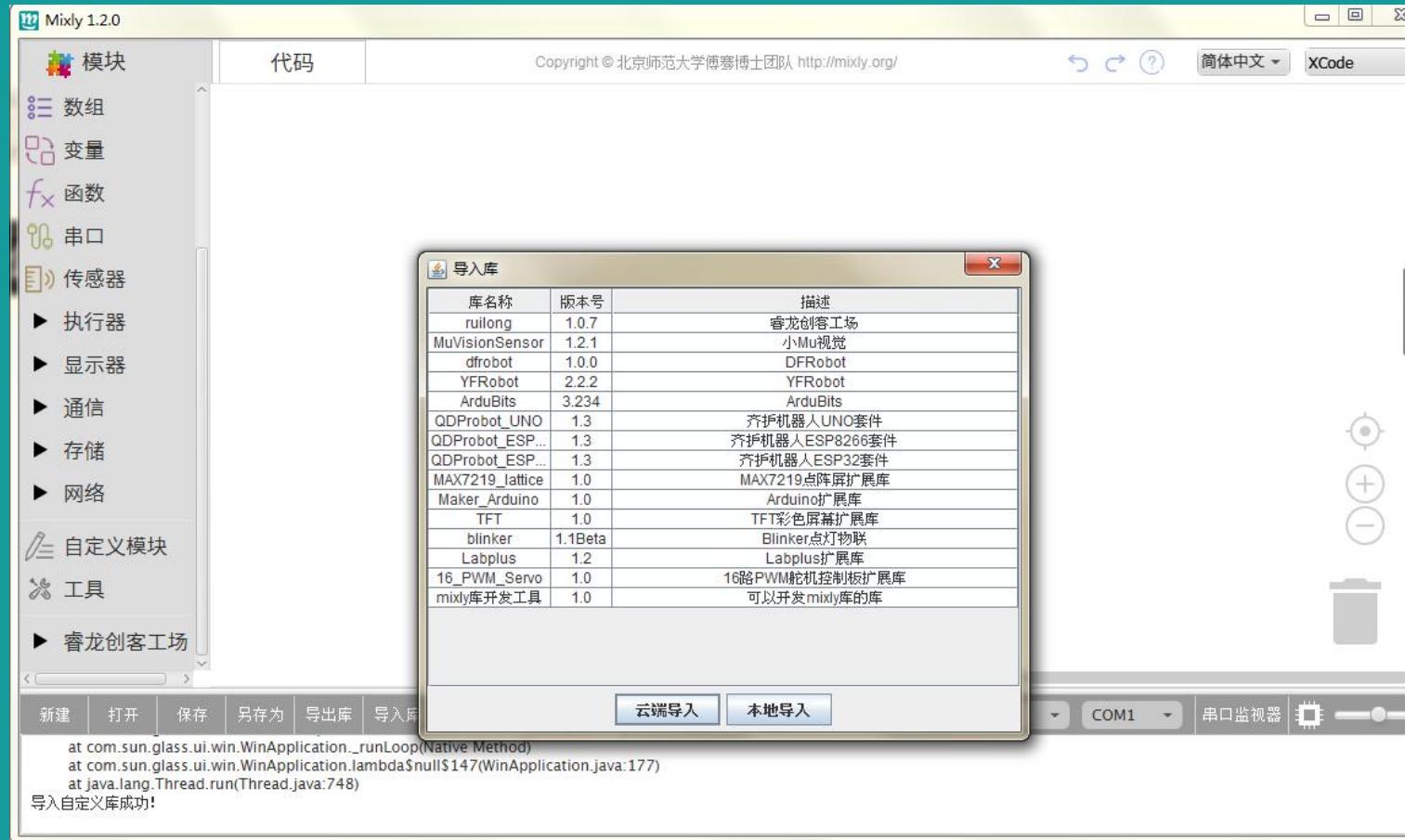
1. Mixly软件安装，软件下载地址:
址:https://mixly.readthedocs.io/zh_CN/latest/basic/02Installation-update.html

2. 驱动程序安装，驱动程序下载：
<http://www.wch.cn/product/CH340.html>

3. 软件帮助文档：
https://mixly.readthedocs.io/zh_CN/latest/



企业库安装



第一步：导入库

第二步：选择ruilong

第三步：云端导入



驱动安装



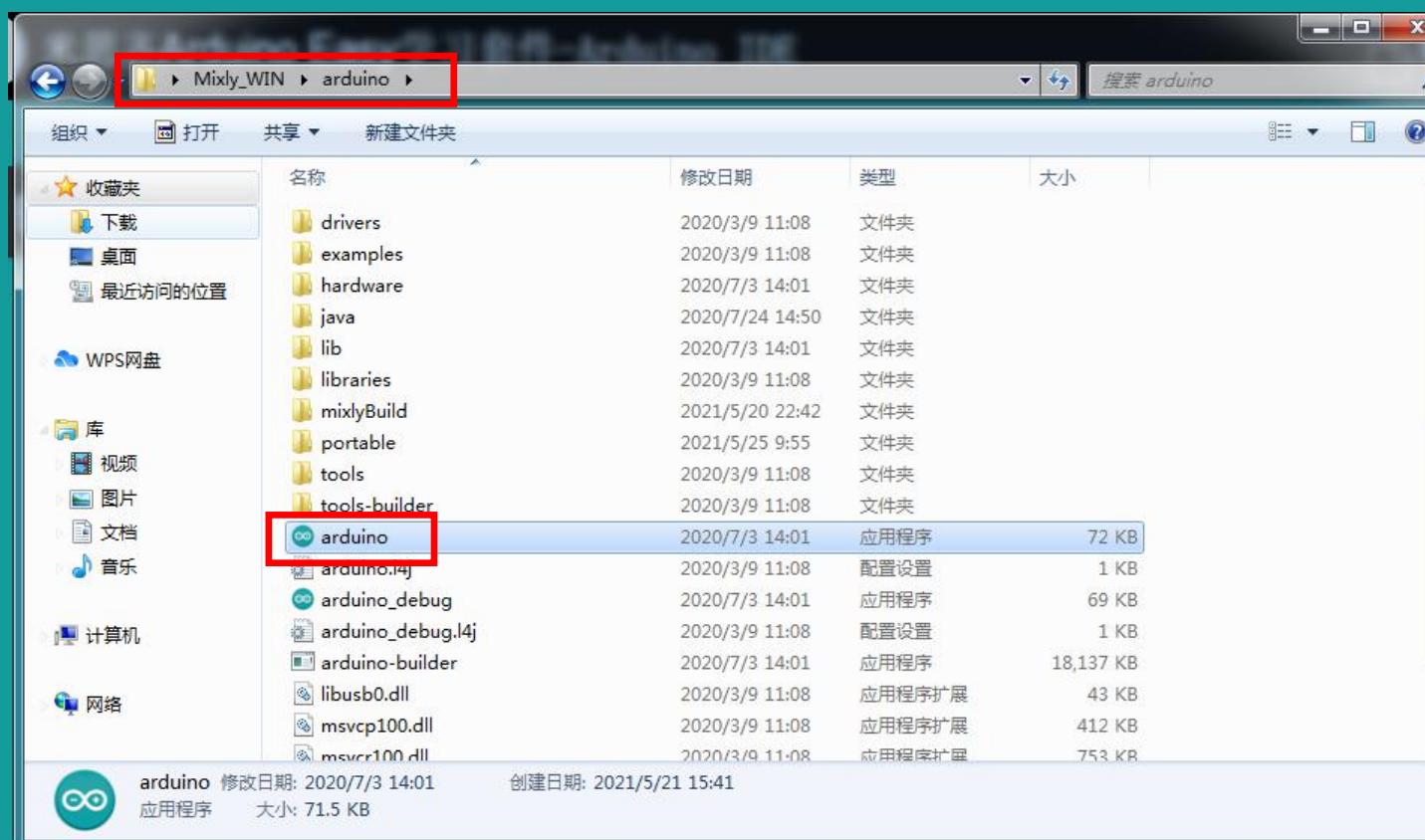
第一步：找到软件安装包，选择arduino目录下的drivers文件夹，选择CH341SE-Driver

第二步：选择CH341SE-Driver文件夹下SETUP.EXE，点击进行安装

第三步：驱动安装成功后进入设备管理器查看端口，本课程端口为USB-SERIAL CH340(COM4),注意不同电脑可能会有区别，以自己电脑为准。



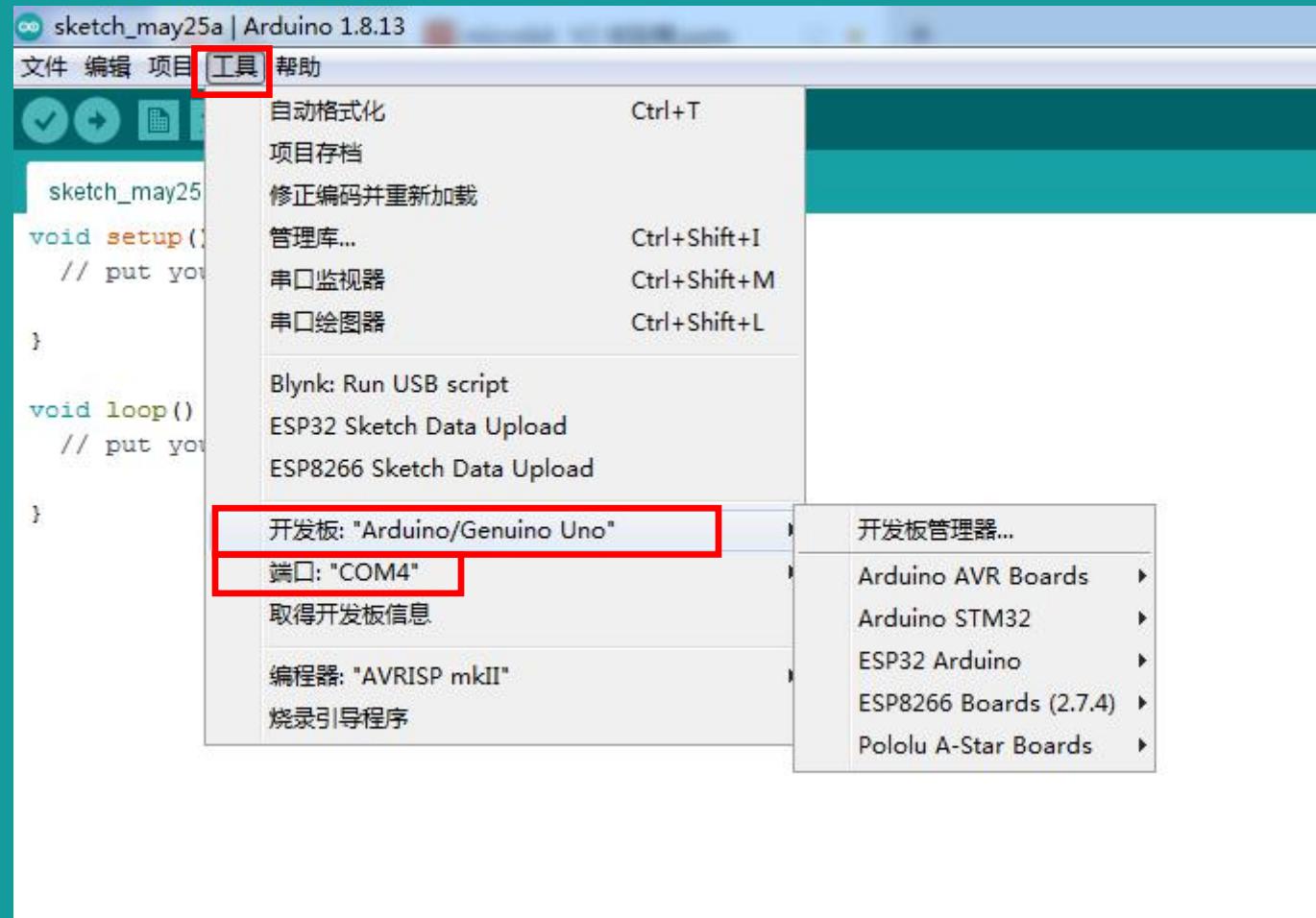
软件介绍



打开下载好的米思齐安装包，打开arduino文件夹，可以找到arduino IDE，利用arduino可以进行代码编程。



软件介绍

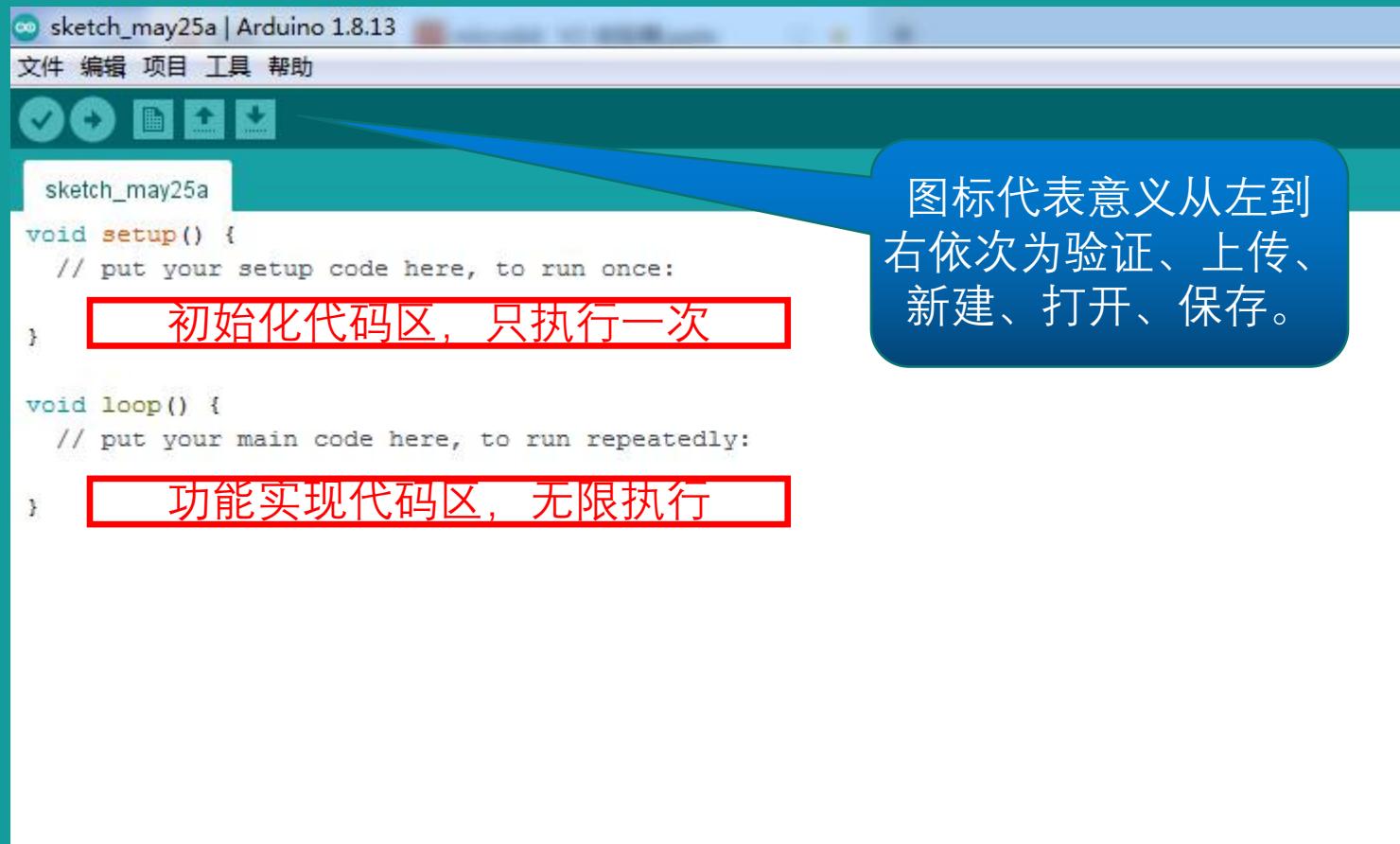


打开arduino IDE 选择工具栏，下拉菜单中可以选择开发板型号。

打开arduino IDE 选择工具栏，下拉菜单选择端口号，本课程端口号为“COM4”



软件介绍





基础传感器应用



项目简析



项目任务

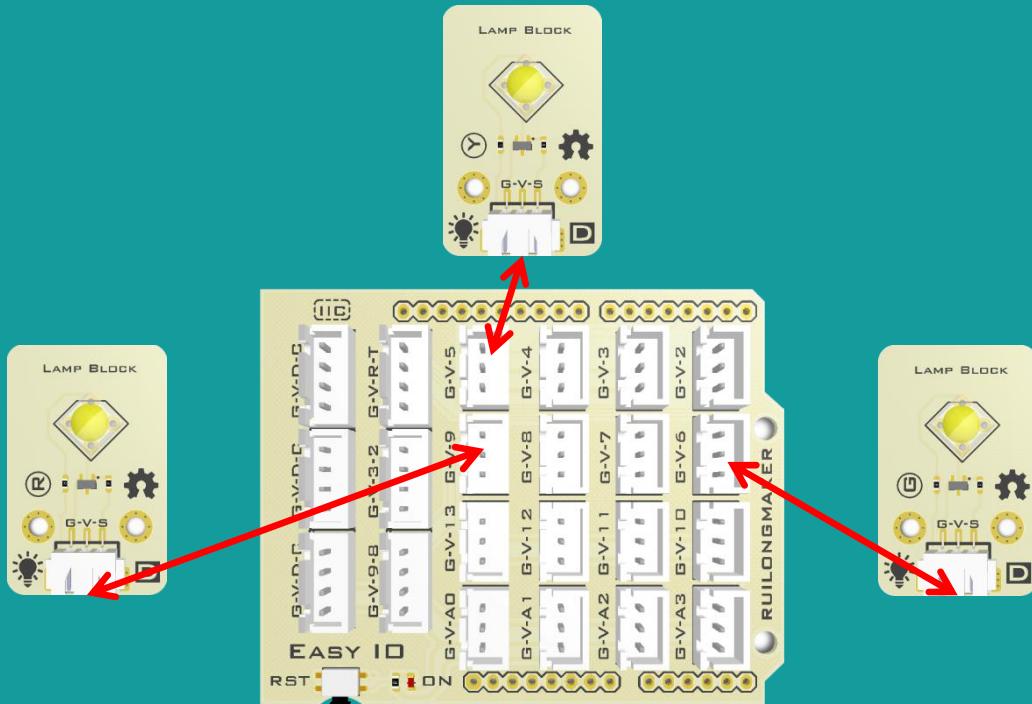
利用闪灯模块（绿色、黄色、红色）实现流水灯效果。

思路分析

闪灯模块默认为低电平，熄灭，当管脚设置为高电平时，闪灯模块亮起。通过延时实现闪灯模块的流水灯效果。



硬件连线



序号	名称	连线	备注
1	闪灯模块 (黄色)	D5	
2	闪灯模块 (绿色)	D6	
3	闪灯模块 (红色)	D6	



程序编程



绿灯亮起，延时1000ms,绿灯熄灭，延时1000ms。

黄灯亮起，延时1000ms,黄灯熄灭，延时1000ms。

红灯亮起，延时1000ms, 红灯熄灭，延时1000ms。



1、setup函数

```
/*通过pinMode函数，可以将Arduino的引脚配置为以下三种模式
1、输出(OUTPUT)模式
当引脚设置为输出 (OUTPUT) 模式时，引脚为低阻抗状态。
这意味着Arduino可以向其它电路元器件提供电流

2、输入(INPUT)模式
当引脚设置为输入 (INPUT) 模式时，引脚为高阻抗状态
此时该引脚可用于读取传感器信号或开关信号。

3、输入上拉 (INPUT_PULLUP) 模式
Arduino 微控制器自带内部上拉电阻。如果你需要使用该内部上拉电阻，
可以通过pinMode()将引脚设置为输入上拉 (INPUT_PULLUP) 模式。
*/
void setup(){
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(9, OUTPUT);
}
```

2、loop函数

```
void loop(){
  digitalWrite(6,HIGH);//向6号管脚写入高电平
  delay(1000);//延时1000ms
  digitalWrite(6,LOW);//向6号管脚写入低电平
  delay(1000);
  digitalWrite(5,HIGH);
  delay(1000);
  digitalWrite(5,LOW);
  delay(1000);
  digitalWrite(9,HIGH);
  delay(1000);
  digitalWrite(9,LOW);
  delay(1000);

}
```



项目简析



项目任务

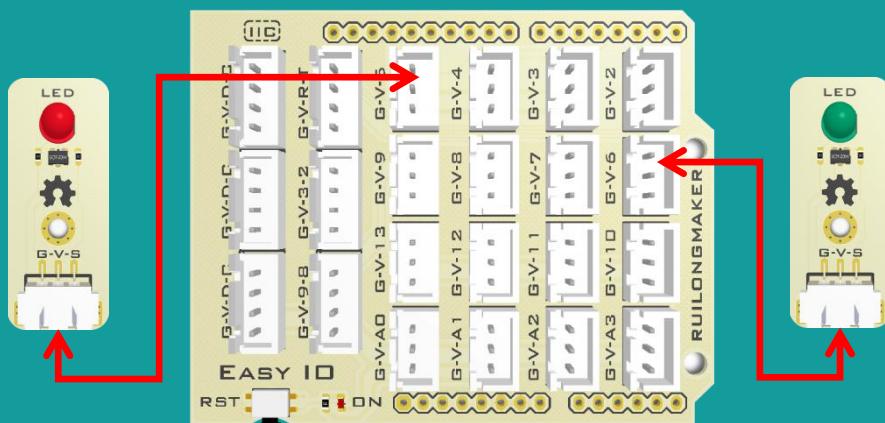
Arduino的3、4、6、10、11管脚具有模拟输出功能，模拟输出的范围为0-255，利用两个LED模块（红色、绿色）实现流光沙漏的效果。

思路分析

要实现流光沙漏的效果，两盏灯的亮度总和应该为255，当其中一盏灯的亮度为x时，另外一盏灯的亮度应该为 $255-x$,使用米思齐控制模块里面的循环语句来实现。



硬件连线



序号	名称	连线	备注
1	LED模块 (红色)	D5	
2	LED模块 (绿色)	D6	



程序编程



```
void setup(){  
}  
void loop(){  
    for (int 亮度 = 0; 亮度 <= 255; 亮度 = 亮度 + (1)) {  
        analogWrite(5,亮度); //向5号管脚写入亮度  
        analogWrite(6,(255 - 亮度)); //向6号管脚写入255-亮度  
        delay(5);  
    }  
    for (int 亮度 = 255; 亮度 >= 0; 亮度 = 亮度 + (-1)) {  
        analogWrite(5,亮度);  
        analogWrite(6,(255 - 亮度));  
        delay(5);  
    }  
}
```

- 1、使用循环结构，逐渐将5号管脚所连灯点亮同时熄灭6号管脚所连灯。
- 2、使用循环结构，逐渐将6号管脚所连灯点亮同时熄灭5号管脚所连灯。



项目简析



项目任务

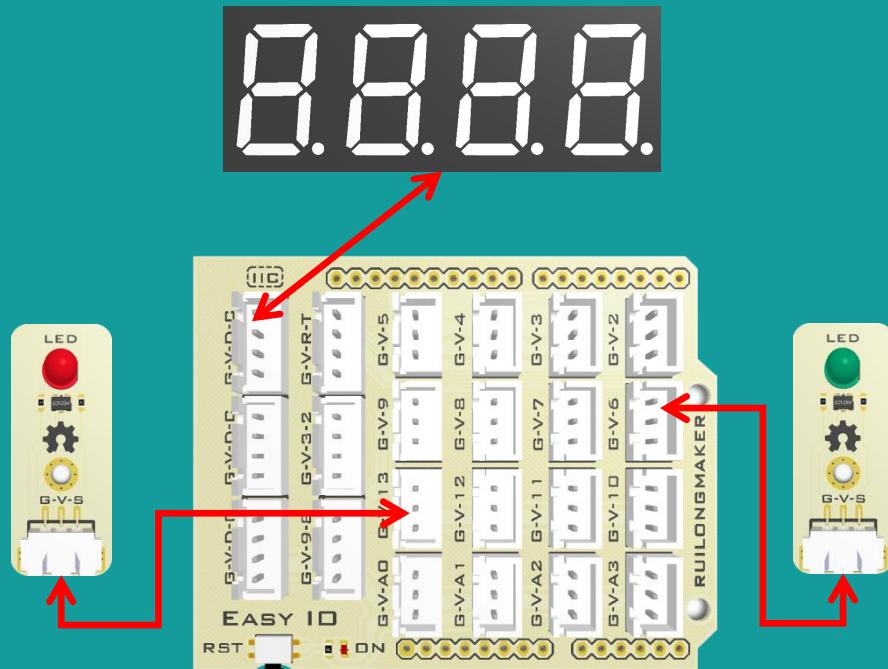
利用TM1650四位数码管、LED模块（红色、绿色）制作一个“计时红黄绿灯”，数码管显示剩余时间。

思路分析

默认绿灯亮起，设置好倒计时时间，四位数码管显示倒计时时间，绿灯亮起结束后，红灯亮起。



硬件连线



序号	名称	连线	备注
1	LED模块 (红色)	D13	
2	LED模块 (绿色)	D6	
3	TM1650数码管模块	GVDC	



程序编程



- 1、默认绿灯亮起。
- 2、设置绿灯亮起时间为30秒，使用循环将倒计时显示在数码管模块上。
- 3、绿灯亮起结束后，红灯亮起。
- 4、红灯亮起时间为60秒，使用循环将倒计时显示在数码管模块上。



1、头文件引入

```
#include <Wire.h>
#include <TM1650.h>
```

2、创建对象

```
TM1650 rUILONG_4display;//创建TM650类的对象
```

3、setup函数

```
void setup(){
    pinMode(5, OUTPUT);//5号管脚设置为输出模式
    digitalWrite(5,HIGH);//向5号管脚写入高电平，绿灯亮
    Wire.begin();
    rUILONG_4display.init();
    pinMode(13, OUTPUT);
}
```

4、loop函数

```
void loop(){
    if (digitalRead(5)) {
        for (int i = 30; i >= 0; i = i + (-1)) {
            delay(1000);
            rUILONG_4display.displayString(i);//四位数码管显示倒计时时间
        }
        digitalWrite(13,HIGH);
        digitalWrite(5,LOW);//5号管脚写入低电平，绿灯灭
    } else if (digitalRead(13)) {
        for (int i = 60; i >= 0; i = i + (-1)) {
            delay(1000);
            rUILONG_4display.displayString(i);
        }
        digitalWrite(5,HIGH);
        digitalWrite(13,LOW);
    }
}
```



项目简析



项目任务

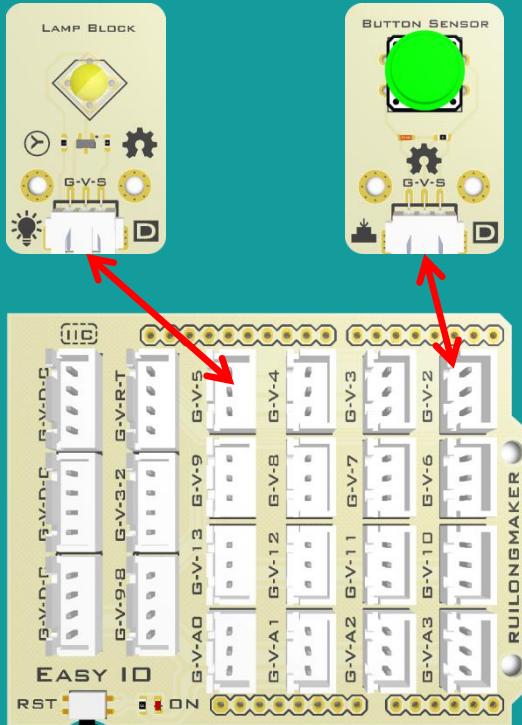
Arduino的3、4、6、10、11管脚具有模拟输出功能，模拟输出的范围为0-255，按键传感器默认抬起为高电平，按下为低电平，利用按钮传感器和一个黄色闪灯模块实现四档调光台灯，按下按钮，档位切换，闪灯模块亮度变化。

思路分析

四档调光台灯有四个档位0, 1, 2, 3，对应亮度从不亮到最亮四个状态，档位用变量来控制，使用取余运算保证档位在0, 1, 2, 3之间切换，最后通过模拟输出调节亮度。



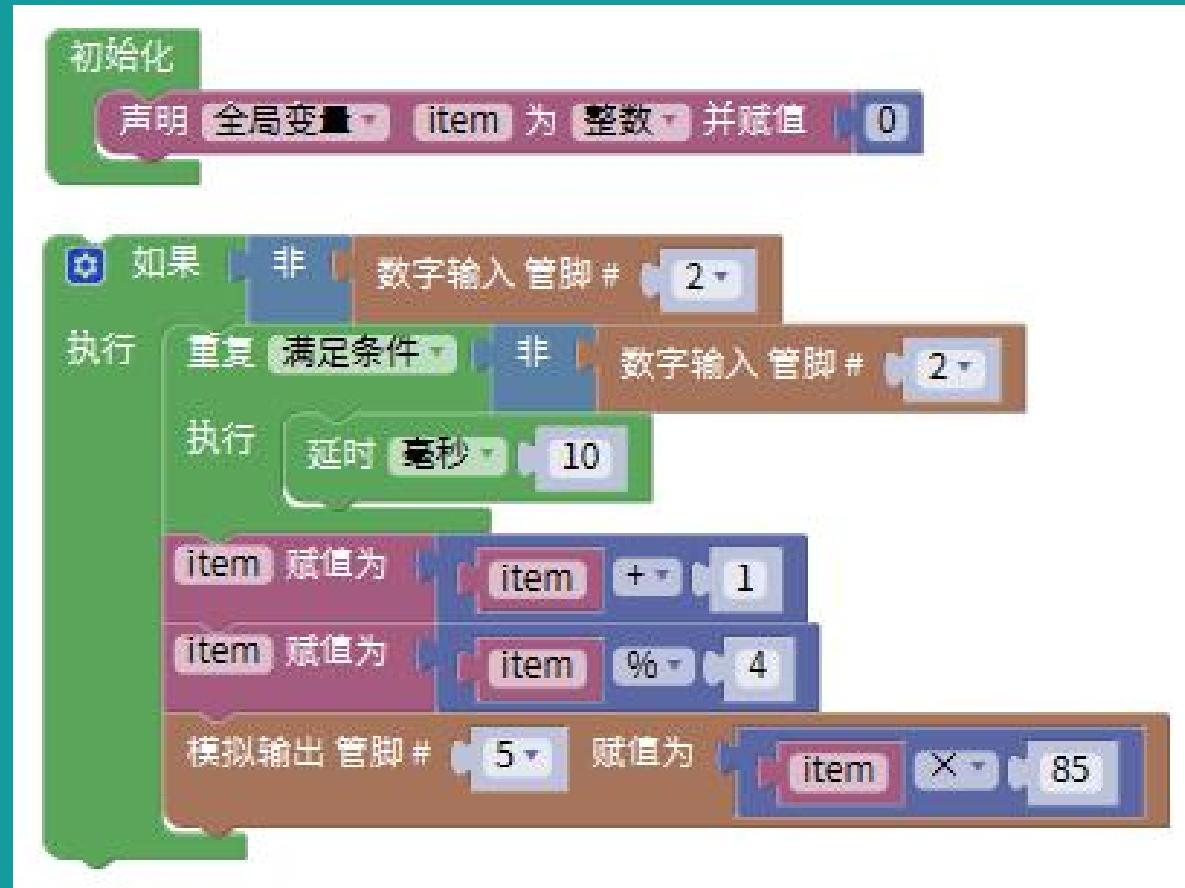
硬件连线



序号	名称	连线	备注
1	按钮传感器	D2	
2	闪灯模块	D5	



程序编程



```
volatile int item;//定义变量
void setup(){
    item = 0;//变量初始化
    pinMode(2, INPUT);//管脚2设为输入模式，高阻抗。
}
void loop(){
    if (!digitalRead(2)) {//默认按钮传感器抬起为高电平（1），按下为低（0）
        while (!digitalRead(2)){
            delay(10); //延时10ms
        }
        item = item + 1; //档位累加
        item = (long) (item) % (long) (4); //取余
        analogWrite(5,(item * 85)); //向5号管脚写入模拟值。
    }
}
```

- 1、按钮传感器默认为高电平（1），按下为低电平（0），重复执行延时10ms的意义为按键消抖。
- 2、使用取余运算保证档位在0、1、2、3之间切换。
- 3、使用模拟输出调节亮度。



项目简析



项目任务

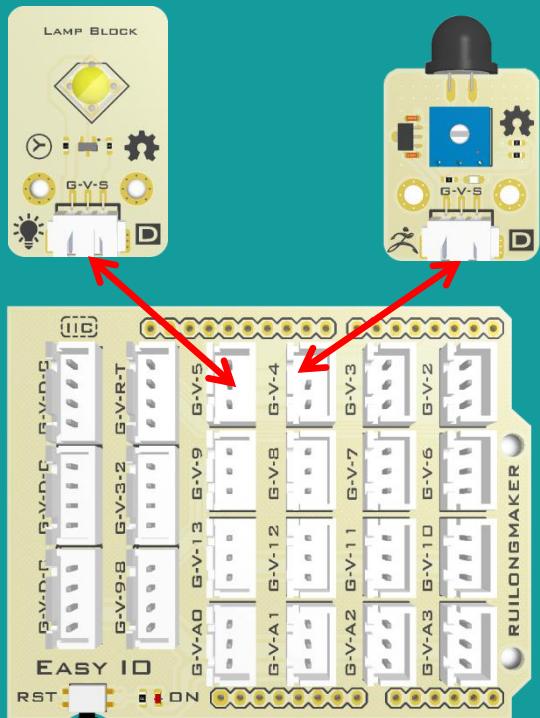
热释电传感器是数字传感器，默认为低电平（0），检测到人体经过后，变为高电平（1），利用热释电传感器和闪灯模块，制作一个人体感应灯，当感受到人体经过时，闪灯模块闪烁。

思路分析

热释电传感器感受到人体经过，变为高电平，向闪灯模块写入读取到的连接闪灯模块管脚电平相反值，延时50ms,每隔50ms闪灯模块实现亮灭切换。



硬件连线



序号	名称	连线	备注
1	热释电传感器	D4	
2	闪灯模块	D5	



程序编程



```
void setup(){
  pinMode(4, INPUT); //4号管脚设置为输入模式
  pinMode(5, OUTPUT); //5号管脚设置为输出模式
}

void loop(){
  if (digitalRead(4)) { //读取4号管脚电平，若为1，则执行
    digitalWrite(5, !digitalRead(5)); //5号管脚写入读取到5号管脚电平相反值
    delay(50); //延时50ms
  }
}
```

- 1、非零即为真，当热释电传感器感受到人体经过时，变为高电平（1）。
- 2、向5号管脚写入读取到5号管脚电平值的相反值。
- 3、延时50ms,5号管脚连接的闪灯模块间隔50ms进行亮灭切换。



项目简析



项目任务

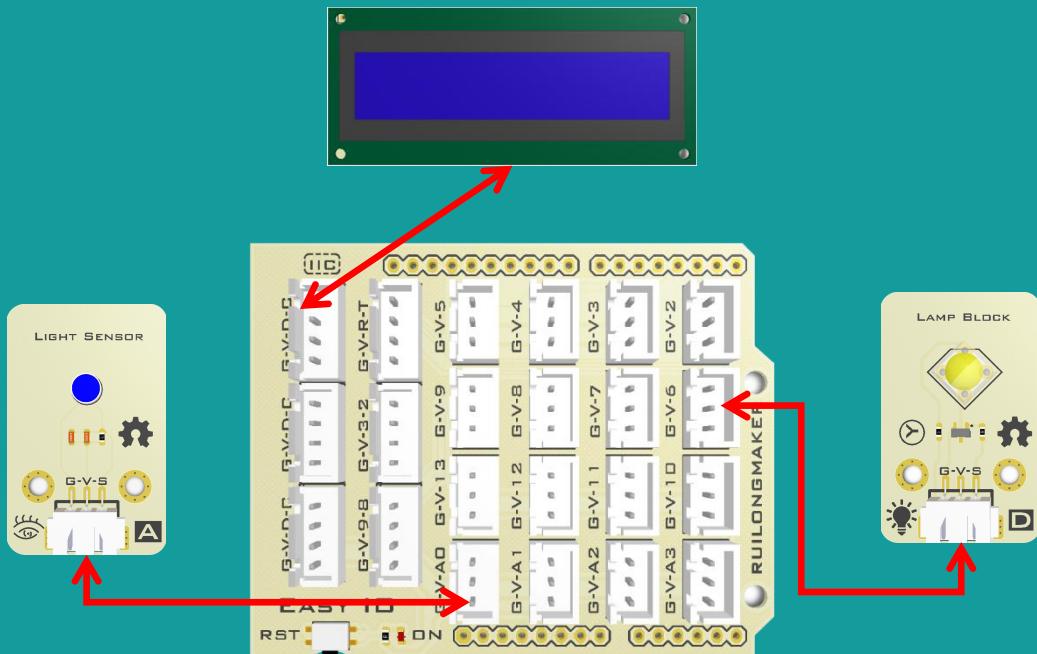
光线传感器是模拟传感器，取值范围是0-1023，利用光线传感器和LCD显示屏、闪灯模块制作一个光线检测器，LCD显示屏显示当前光线强度，闪灯模块依据检测到的光线强度显示不同亮度。

思路分析

初始化变量保存光线传感器检测到的当前光线强度并且取整，串口打印当前光线强度，LCD显示屏显示当前光线强度，建立光线强度和闪灯模块亮度的映射关系，根据光线强度调节闪灯模块亮度。



硬件连线



序号	名称	连线	备注
1	光线传感器	A0	
2	LCD显示屏	GVDC	
3	闪灯模块	D6	



程序编程



- 1、初始化变量用于保存光线传感器获取到的模拟值。
- 2、串口打印模拟值。
- 3、LCD显示屏显示光线传感器获取到的模拟值。
- 4、建立光线传感器模拟值和闪灯模块亮度模拟值的映射关系，以调节闪灯模块亮度。



1、头文件引入

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

2、定义变量

```
volatile int guangxian;
```

3、创建对象

```
LiquidCrystal_I2C mylcd(0x27,16,2); //创建mylcd对象并传入三个实参
```

4、setup函数

```
void setup(){
    guangxian = 0;
    Serial.begin(9600);
    mylcd.init();
    mylcd.backlight();
}
```

5、loop函数

```
void loop(){
    guangxian = analogRead(A0); //获取当前光线强度模拟值
    Serial.println(guangxian); //串口打印光线强度
    mylcd.setCursor(0, 0);
    mylcd.print("Hi..ruilongmaker"); //LCD显示屏第一行打印
    mylcd.setCursor(0, 1);
    mylcd.print("");
    mylcd.setCursor(1-1, 2-1);
    mylcd.print(guangxian); //LCD显示屏第二行打印
    analogWrite(6,(map(guangxian, 0, 1023, 0, 255))); //建立光线强度和闪灯模块亮度映射关系并且写入6号管脚
    delay(1000); //延时
    mylcd.clear(); //清屏
}
```



项目简析



项目任务

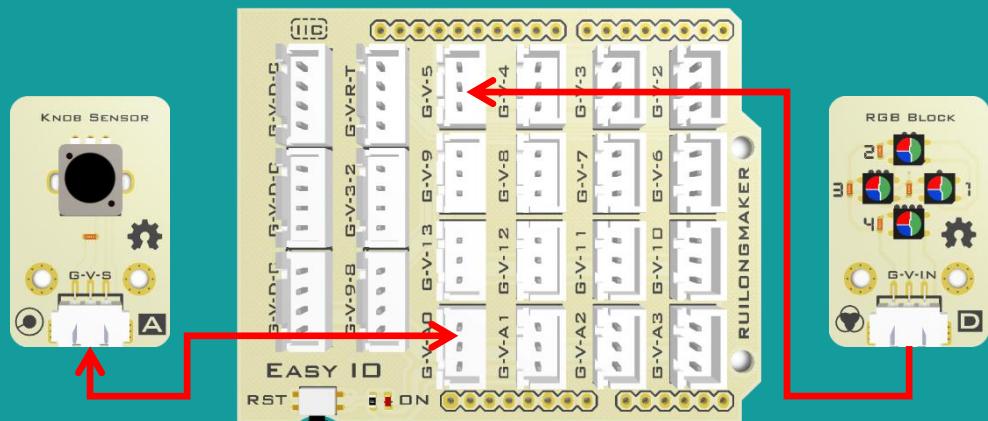
Arduino的A0、A1、A2、A3、A4、A5具有模拟输入的功能，也可以作为数字输入或输出，旋钮传感器输入的是模拟信号。利用旋钮传感器和RGB彩灯模块，制作一个可调节RGB彩灯。

思路分析

初始化彩灯模块，连接管脚为D5，灯数为4。串口打印旋钮传感器模拟输入值。映射模拟输入的值，模拟输入的范围为0-1023，模拟输出的范围是0-255。转动旋钮传感器，可以实现对RGB彩灯模块亮度的调节。



硬件连线



序号	名称	连线	备注
1	旋钮传感器	A0	
2	RGB彩灯模块	D5	



程序编程



- 1、初始化彩灯模块，连接管脚为D5，灯数为4。
- 2、串口打印旋钮传感器模拟输入值。
- 3、映射模拟输入的值，模拟输入的范围为0-1023，模拟输出的范围是0-255。



1、头文件引入

```
#include <Adafruit_NeoPixel.h>
```

2、loop函数

```
//构造体，参数4代表RGB灯数，5代表管脚编号，NEO_GRB+NEO_KHZ800,代表RGB类型。  
Adafruit_NeoPixel rgb_display_5(4,5, NEO_GRB + NEO_KHZ800);  
uint32_t Wheel(byte WheelPos){  
    if(WheelPos < 85){  
        return rgb_display_5.Color(WheelPos * 3, 255 - WheelPos * 3, 0);  
    }  
    else if(WheelPos < 170){  
        WheelPos -= 85;  
        return rgb_display_5.Color(255 - WheelPos * 3, 0, WheelPos * 3);  
    }  
    else{  
        WheelPos -= 170;return rgb_display_5.Color(0, WheelPos * 3, 255 - WheelPos * 3);  
    }  
}
```

3、setup函数

```
void setup(){  
    rgb_display_5.begin();  
    Serial.begin(9600);  
}
```

4、loop函数

```
void loop(){  
    Serial.println(analogRead(A0));  
    for(int RGB_RAINBOW_i = 0; RGB_RAINBOW_i < rgb_display_5.numPixels(); RGB_RAINBOW_i++){  
        rgb_display_5.setPixelColor(RGB_RAINBOW_i, Wheel(((RGB_RAINBOW_i * 256 /  
rgb_display_5.numPixels()) + (map(analogRead(A0), 0, 1023, 0, 255))) & 255));  
    }  
    rgb_display_5.show();  
    delay(10);  
}
```



项目简析



项目任务

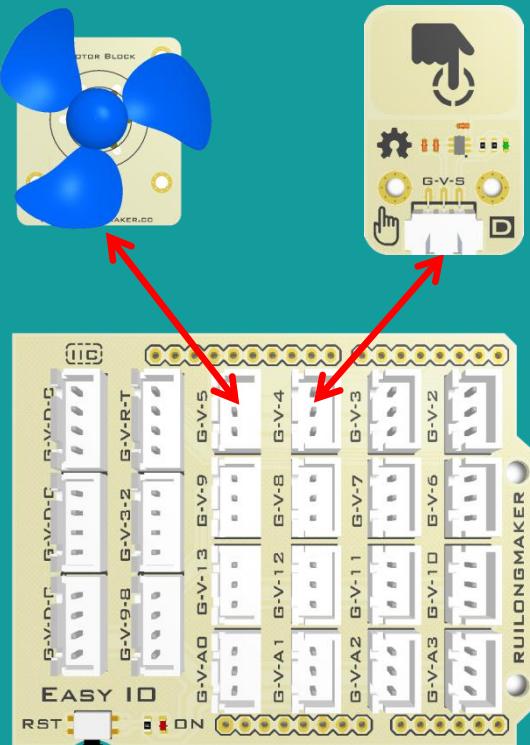
触摸传感器默认为低电平（0），当按下触摸传感器，变为高电平（1），利用触摸传感器和马达风扇模块制作一个触摸风扇。

思路分析

定义变量用于对风扇速度的切换，使用取余运算保证档位在0、1、2、3之间切换。风扇连接模拟输出管脚，以实现对风扇速度的调节。



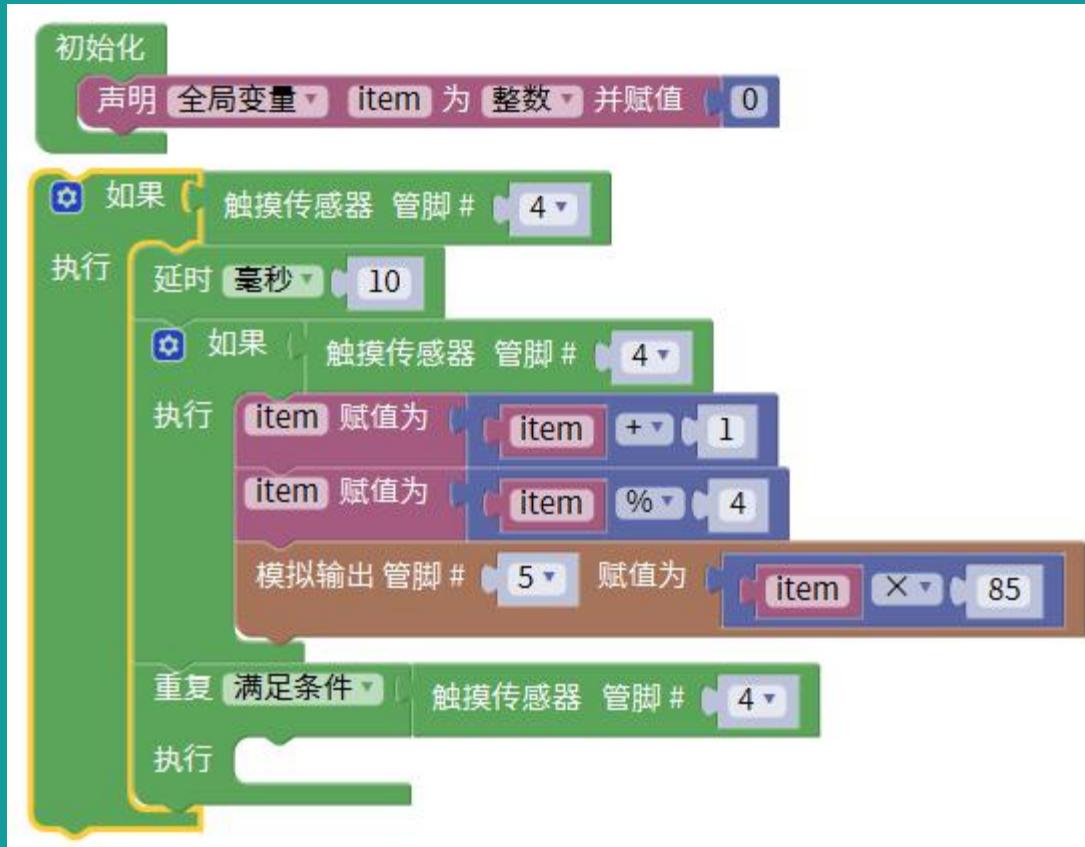
硬件连线



序号	名称	连线	备注
1	触摸传感器	D4	
2	马达风扇模块	D5	



程序编程



```
volatile int item;//定义变量
void setup(){
  item = 0;
  pinMode(4, INPUT);//管脚4设置为输入模式
}
void loop(){
  if (digitalRead(4)) {//读取4号管脚电平
    delay(10);
    if (digitalRead(4)) {
      item = item + 1;
      item = (long) (item) % (long) (4);//取余运算
      analogWrite(5,(item * 85));//向5号管脚写入模拟值
    }
  while (digitalRead(4)) {
    }}}
```

- 1、触摸传感器默认为低电平（0），按下为高电平（1）
- 2、使用取余运算保证档位在0、1、2、3之间切换。
- 3、使用模拟输出调节亮度。



项目简析



项目任务

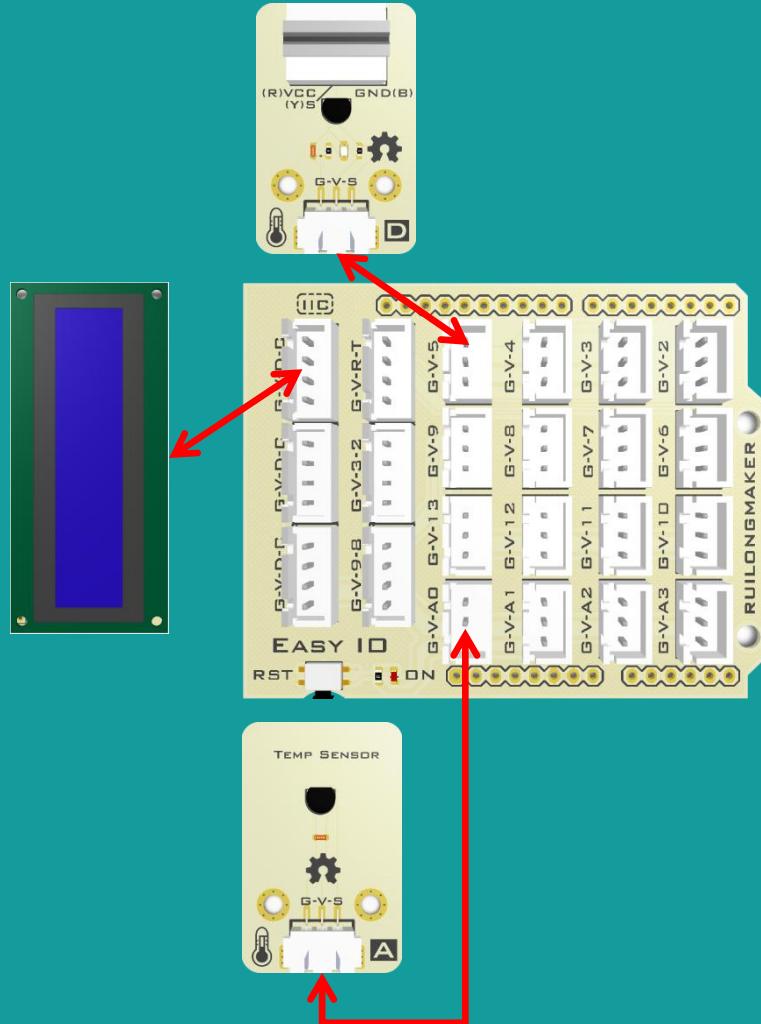
LM35温度传感器是一种半导体温度传感器，采用模拟信号输出，其测量是0-100摄氏度。DS18B20是常用的数字传感器，-55摄氏度至+125摄氏度。华氏相当于是-67到257。利用两个不同的温度传感器、LCD显示屏，制作一个温度测量器，LCD显示屏第一行显示摄氏度，第二行能显示华氏摄氏度。

思路分析

初始化LCD显示屏，利用串口分别打印LM35温度传感器和DS18B2O温度传感器数值，显示屏第一行显示LM35温度，显示屏第二行显示DS18B2O温度。



硬件连线



序号	名称	连线	备注
1	LCD显示屏	GVDC	
2	LM35温度传感器	A0	
3	DS18B20温度传感器	D5	



程序编程



- 1、初始化LCD显示屏
- 2、串口分别打印温度值，在串口监视器可以查询数值。
- 3、LCD显示屏第一行打印LM35温度值。
- 4、LCD显示屏第二行打印DS18B20华氏温度值。



1、头文件引入

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <OneWire.h>
#include <DallasTemperature.h>
```

2、结构体变量

```
LiquidCrystal_I2C mylcd(0x27,16,2);
OneWire oneWire_5(5);
DallasTemperature sensors_5(&oneWire_5);
DeviceAddress insideThermometer;
```

3、DS18B20获取温度函数

```
float ds18b20_5_getTemp(int w) {
    sensors_5.requestTemperatures();
    if(w==0) {
        return sensors_5.getTempC(insideThermometer);}
    else {
        return sensors_5.getTempF(insideThermometer);}}
```

4、setup函数

```
void setup(){
    mylcd.init();
    mylcd.backlight();
    Serial.begin(9600);
    sensors_5.getAddress(insideThermometer, 0);
    sensors_5.setResolution(insideThermometer, 9);}
```

5、loop函数

```
void loop(){
    Serial.println(analogRead(A0)*0.488);
    Serial.println(ds18b20_5_getTemp(1));
    mylcd.setCursor(1-1, 1-1);
    mylcd.print(String("LM35:") + String(String(analogRead(A0)*0.488) + String(".C")));
    mylcd.setCursor(1-1, 2-1);
    mylcd.print(String("DS18:") + String(String(ds18b20_5_getTemp(1)) + String(".C")));
    delay(1000);}
```



项目简析



项目任务

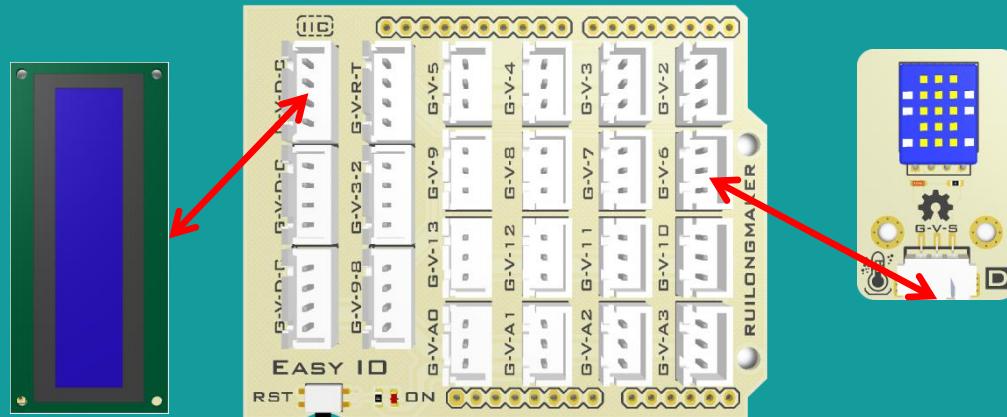
DTH11温湿度传感器，精度湿度 $\pm 5\%$ RH,温度 $\pm 2^{\circ}\text{C}$,量程湿度20-90%RH,温度测量范围0~50°C。利用温湿度传感器和LCD显示屏显示温度和湿度。

思路分析

初始化LCD显示屏，利用串口分别打印DTH11温湿度传感器温度和湿度，LCD显示屏第一行显示温度，第二行显示湿度。



硬件连线



序号	名称	连线	备注
1	LCD显示屏	GVDC	
2	DTH11温湿度传感器	D6	



程序编程



- 1、初始化LCD显示屏
- 2、串口分别打印温湿度传感器检测温度值，检测湿度值。
- 3、LCD显示屏第一行打印温度值。
- 4、LCD显示屏第二行打印湿度值。



1、头文件引入

```
#include <dht11.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

2、结构体变量

```
LiquidCrystal_I2C mylcd(0x27,16,2);
dht11 myDHT_6;
```

3、获取温度函数

```
int dht_6_gettemperature() {
    int chk = myDHT_6.read(6);
    int value = myDHT_6.temperature;
    return value;}
```

4、获取湿度函数

```
int dht_6_gethumidity() {
    int chk = myDHT_6.read(6);
    int value = myDHT_6.humidity;
    return value;}
```

5、setup函数

```
void setup(){
    mylcd.init();
    mylcd.backlight();
    Serial.begin(9600);}
```

6、loop函数

```
void loop(){
    Serial.println(dht_6_gettemperature());
    Serial.println(dht_6_gethumidity());
    mylcd.setCursor(1-1, 1-1);
    mylcd.print(String("temp:") + String(String(dht_6_gettemperature()) + String(".c")));
    mylcd.setCursor(1-1, 2-1);
    mylcd.print(String("humi:") + String(String(dht_6_gethumidity()) + String("%")));
}
```



项目简析



项目任务

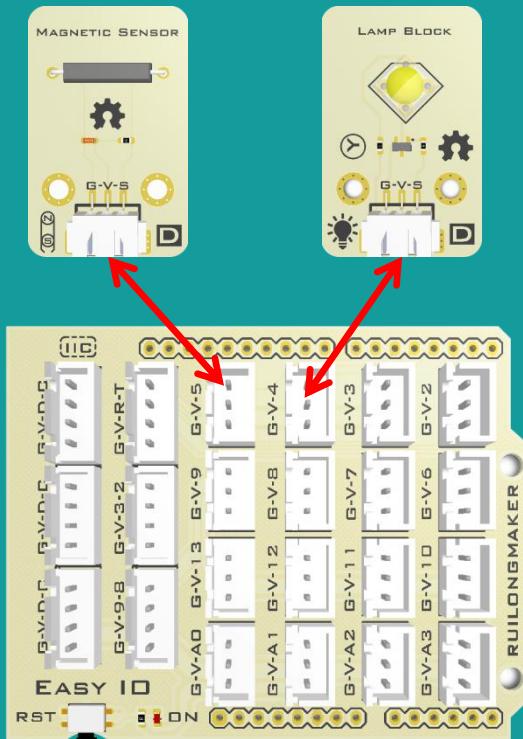
磁敏传感器默认为高电平（1），当检测到带有磁性的物体时，变为低电平（0），利用一个磁敏传感器和闪灯模块制作一个磁性探测器。

思路分析

磁性传感器默认为高电平（1），检测到带有磁性的物体时变为低电平（0），对电平值进行取反，如果检测到磁性物体，闪灯模块亮起，否则闪灯模块熄灭。



硬件连线



序号	名称	连线	备注
1	闪灯模块	D4	
2	磁敏传感器	D5	



程序编程



```
void setup(){
    pinMode(5, INPUT); //管脚5设置为输入模式
    Serial.begin(9600);
    pinMode(4, OUTPUT); //管脚4设置为输出模式
}
void loop(){
    Serial.println(digitalRead(5)); //打印读取磁敏传感器
    if (!digitalRead(5)) { //取反
        digitalWrite(4,HIGH); //4号管脚写入高电平
    } else {
        digitalWrite(4,LOW); //4号管脚写入低电平
    }
}
```

- 1、串口打印磁敏传感器数值，打开串口监视器，默认为1，检测到磁性为0。
- 2、对磁敏传感器电平进行取反。
- 3、如果磁敏传感器检测到磁性，闪灯模块亮起，否则，闪灯模块熄灭。



项目简析



项目任务

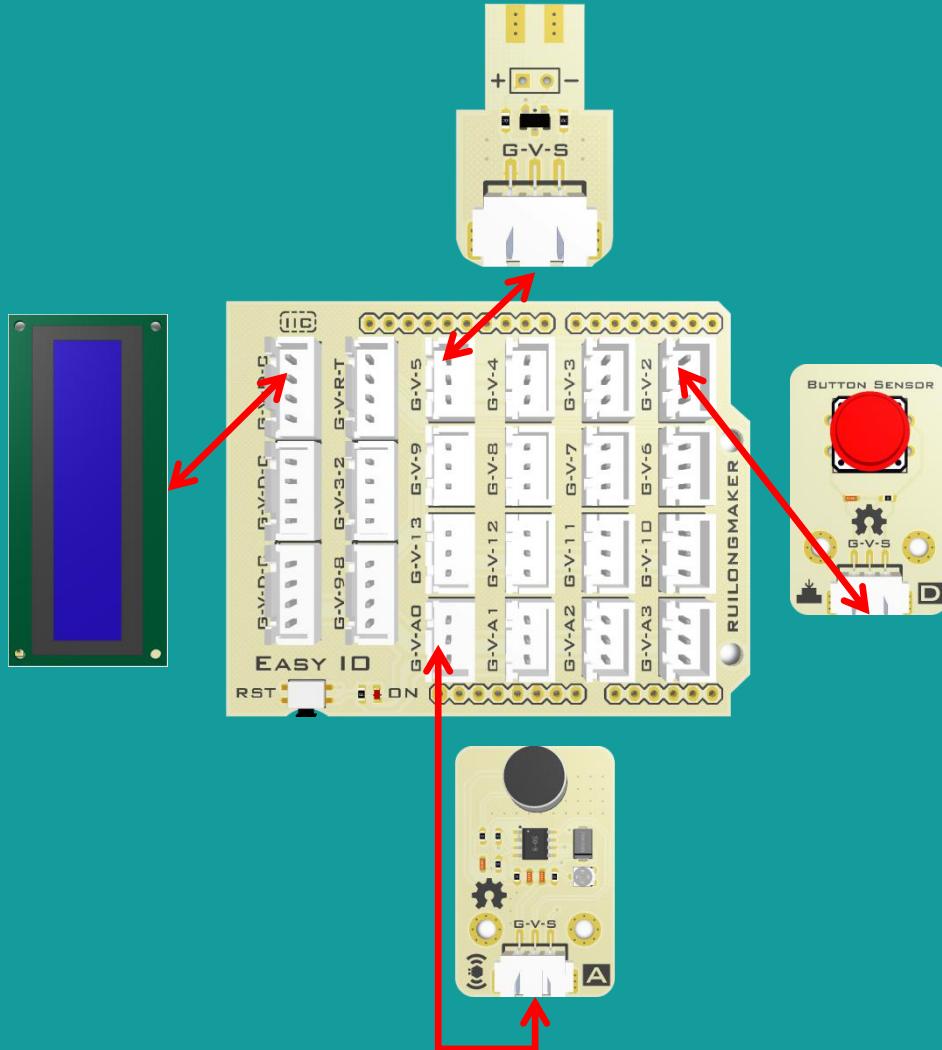
利用LCD显示器模块、LED灯串模块、声音传感器、按钮传感器制作一个声音检测器。

思路分析

按键传感器连接D2管脚，设置软件中断，用于声音检测器的开关，声音传感器是模拟传感器，通过串口打印模拟数值，在串口监视器中可查看，LED灯串模块连接D5管脚，进行模拟输出，将声音传感器的模拟数值和D5管脚的模拟输出进行映射，当按下按钮时，声音越大，LED灯串越亮。



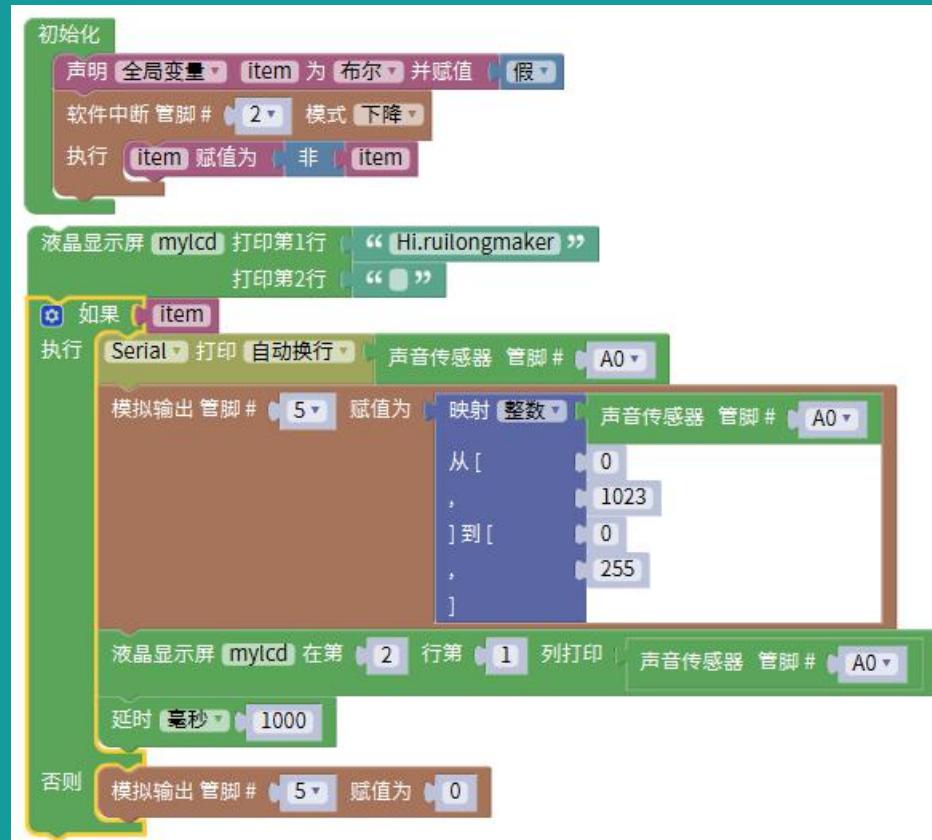
硬件连线



序号	名称	连线	备注
1	LCD显示器	GVDC	
2	按钮传感器	D2	
3	灯串模块	D5	
4	声音传感器	A0	



程序编程



- 1、初始化状态变量item为假，用于切换声音检测器的开关。
- 2、管脚2设置为软件中断模式，按钮传感器默认为高电平，按下为低电平，模式选择下降。item赋值为非item。
- 3、串口打印声音传感器模拟数值。
- 4、映射传感器模拟数值。
- 5、LCD显示屏显示声音传感器数值



1、头文件引入

```
#include <PinChangeInt.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

2、定义变量

```
volatile boolean item;
LiquidCrystal_I2C mylcd(0x27,16,2);
```

3、软件中断函数

```
void attachPinInterrupt_fun_2() {
    item = !item;}
```

4、setup函数

```
void setup(){
    item = false;
    pinMode(2, INPUT);
    PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,FALLING);
    mylcd.init();
    mylcd.backlight();
    Serial.begin(9600);}
```

5、loop函数

```
void loop(){
    mylcd.setCursor(0, 0);
    mylcd.print("Hi.ruilongmaker");
    mylcd.setCursor(0, 1);
    mylcd.print("");
    if (item) {
        Serial.println(analogRead(A0));
        analogWrite(5,(map(analogRead(A0), 0, 1023, 0, 255)));
        mylcd.setCursor(1-1, 2-1);
        mylcd.print(analogRead(A0));
        delay(1000);
    } else {
        analogWrite(5,0); }}
```



项目简析



项目任务

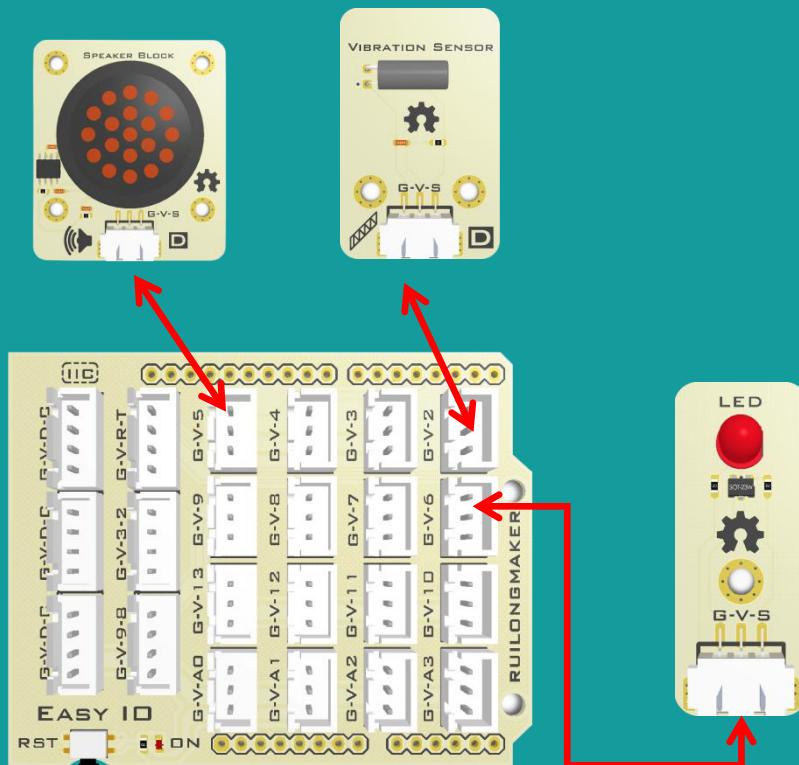
利用震动传感器、LED模块（红色）、喇叭扬声器模块，制作一个入侵报警器。

思路分析

串口打印震动传感器值，打开串口监视器，默认状态下为高电平（1），检测到震动后变为低电平（0），设置软件中断，当检测到震动时，红色LED模块亮起，喇叭扬声器模块播放，进行示警。



硬件连线



序号	名称	连线	备注
1	震动传感器	D2	
2	喇叭扬声器模块	D5	
	LED模块	D6	



程序编程



- 1、串口打印震动传感器数值，默认状态下为高电平（1），检测到震动，变为低电平（0）
- 2、声明变量并赋值为0，设置管脚2为软件中断，模式选择下降，管脚2所连震动传感器检测到震动后，item赋值为1。
- 3、如果检测到震动，红色LED灯亮起，喇叭扬声器模块播放音频，持续时间2000ms(自行设置)
- 4、报警结束后，item赋值为0。



1、头文件引入

```
#include <PinChangeInt.h>
```

2、定义变量

```
volatile boolean item;//布尔  
float period;//浮点  
float pulse;
```

3、软件中断函数

```
void attachPinInterrupt_fun_2(){  
    item = 1;  
}
```

4、newtone函数

```
void newtone(int tonePin, int frequency, int duration){  
    float period = 1000000.0 /frequency;  
    float pulse = period / 2.0;  
    for (int i=1; i<=((duration * 1000.0)/period);i=i+1){  
        pinMode(tonePin, OUTPUT);  
        digitalWrite(tonePin,HIGH);  
        delayMicroseconds(pulse);  
        pinMode(tonePin, OUTPUT);  
        digitalWrite(tonePin,LOW);  
        delayMicroseconds(pulse);}  
}
```

5、setup函数

```
void setup(){  
    pinMode(2, INPUT);  
    Serial.begin(9600);  
    item = 0;  
    //类外函数，参数分别为管教号，软件中断，软件中断模式  
    PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,FALLING);  
    pinMode(6, OUTPUT);  
    pinMode(5, OUTPUT);  
}
```

5、loop函数

```
void loop(){  
    Serial.println(digitalRead(2));//串口打印管脚2电压值  
  
    if (item == 1){  
        digitalWrite(6,HIGH);//向管脚6写入高电平  
        newtone(5,532,2000);//参数5代表5号管脚， 532代表声音频率， 2000代表持续时间  
        digitalWrite(6,LOW);//向管脚6写入低电平  
    }  
    item = 0;  
}
```



项目简析



项目任务

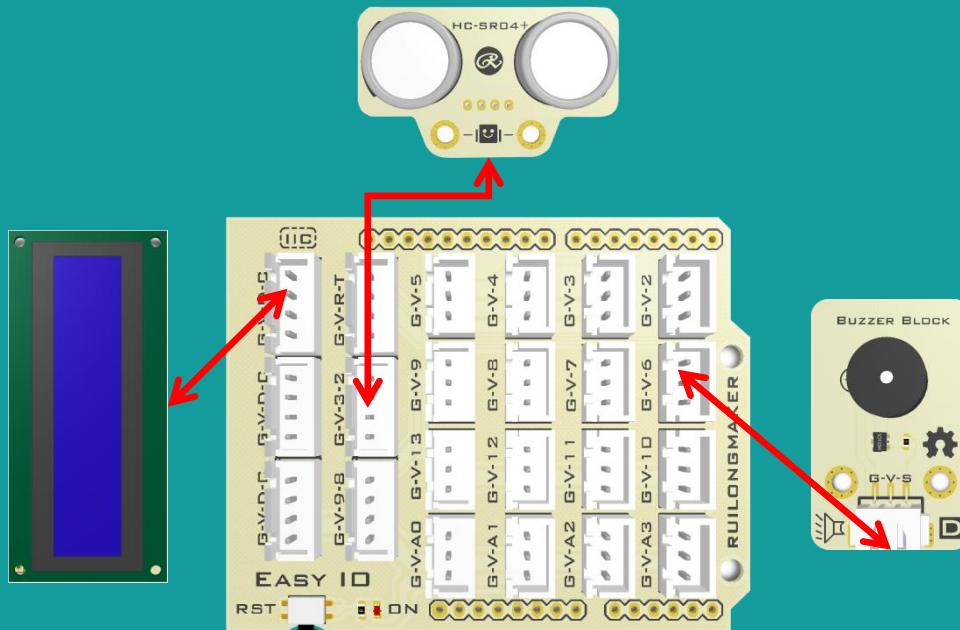
利用LCD显示器模块、蜂鸣器模块、超声波传感器制作一个模拟倒车雷达。

思路分析

定义变量distance用于保存超声波传感器数值，如果distance小于50，进行数学映射，在LCD显示屏上显示超声波传感器测得距离数值，蜂鸣器模块开始报警，距离越小，报警时间间隔越短。



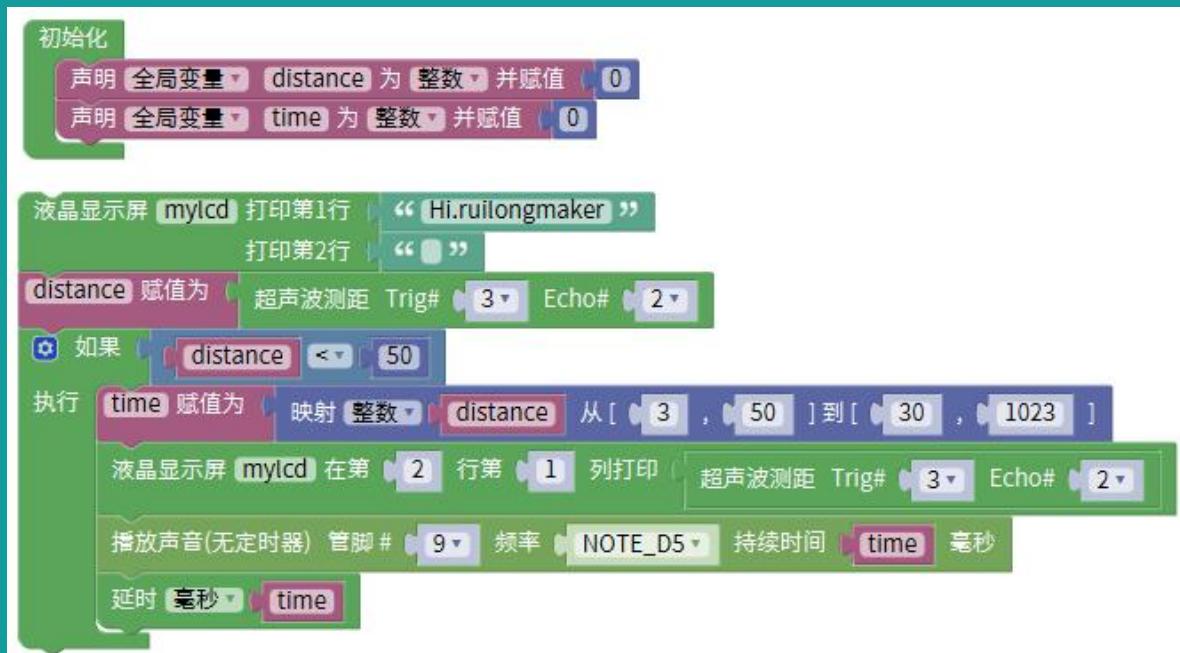
硬件连线



序号	名称	连线	备注
1	LCD显示器	GVDC	
2	超声波传感器	GV32	
3	蜂鸣器模块	D6	



程序编程



- 1、初始化状态变量distance、time
- 2、distance赋值为超声波传感器测得距离数值。
- 3、如果distance<50，将distance进行映射，并赋值给time。
- 4、在LCD显示屏上显示超声波传感器测得数值。
- 5、蜂鸣器开始报警，距离越小，报警间隔越小。



1、头文件引入

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <NewTone.h>
```

2、定义变量

```
volatile int distance;
volatile int time;
```

3、超声波测距函数

```
/*
pulseIn函数是一个简单的测量脉冲宽度的函数，默认单位是us,pulseIn测出来的是超声波从发射到接收所经过的时间,声速大约为343米/秒，即29.15us/cm,发送后接收到回波，要除以58。
*/
float checkdistance_3_2() {
    digitalWrite(3, LOW);//3号管脚写入低电平
    delayMicroseconds(2);//延时2s
    digitalWrite(3, HIGH);//3号管脚写入高电平
    delayMicroseconds(10);//延时10秒
    digitalWrite(3, LOW);//3号管脚写入低电平
    float distance = pulseIn(2, HIGH) / 58.00;
    delay(10);
    return distance;}
```

4、setup函数

```
void setup(){
    distance = 0;
    time = 0;
    mylcd.init();
    mylcd.backlight();
    pinMode(3, OUTPUT);
    pinMode(2, INPUT);
    pinMode(6, OUTPUT);}
```

5、loop函数

```
void loop(){
    mylcd.setCursor(0, 0);
    mylcd.print("Hi.rui long maker");
    mylcd.setCursor(0, 1);
    mylcd.print("");
    distance = checkdistance_3_2();
    if (distance < 50) {
        time = (map(distance, 3, 50, 30, 1023));
        mylcd.setCursor(1-1, 2-1);
        mylcd.print(checkdistance_3_2());
        NewTone(6,587,time);
        delay(time);}}
```



项目简析



项目任务

红外遥控被广泛的应用在家用电器和工业控制等，由发射和接收两部分组成，遥控器发出调制过的红外光波，接收器将遥控器发射的红外光转化为相应的电信号。利用红外接收传感器和闪灯模块，制作一个可以通过红外遥控点亮的灯。按下按键，闪灯模块亮起。

思路分析

Arduino主控板已经集成了一个红外接收传感器，本套件中搭载了一个单独的红外接收传感器，可以用于外接。红外遥控器的每个按键对应了不同的编码，不同的遥控器编码也不相同，可以通过串口监视器打印不同按键对应的编码值，判断该编码值要对应执行的语句，就可以轻松的实现红外遥控。



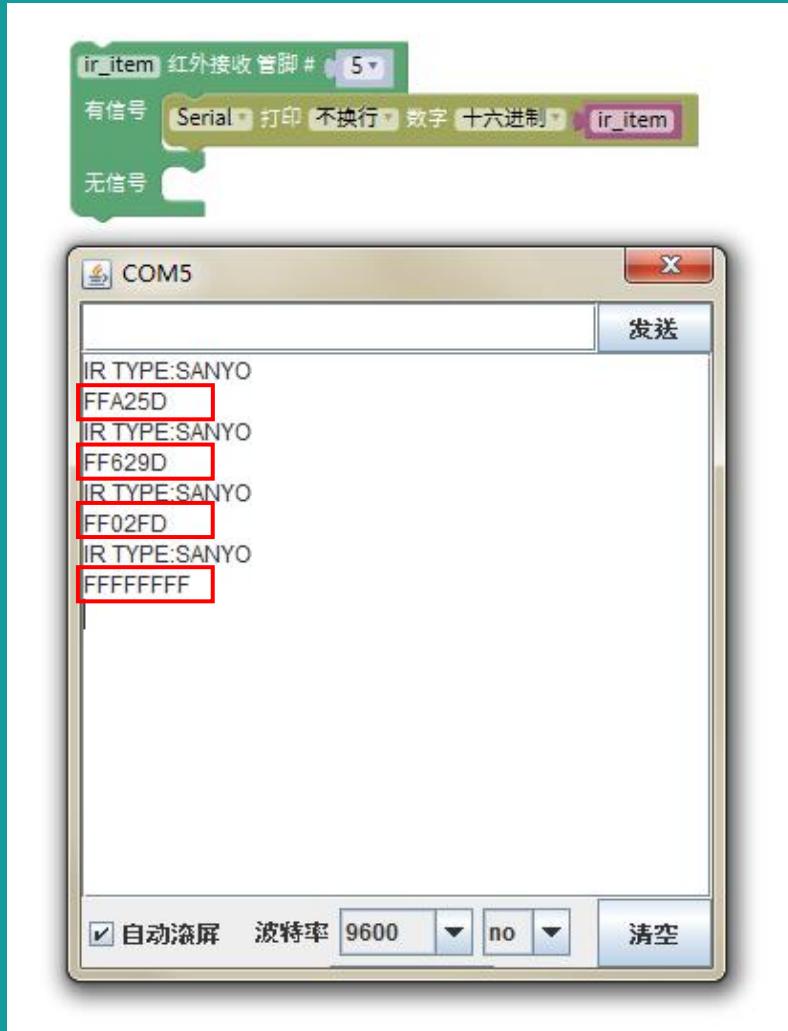
知识接入



- 1、在“通信”类模块中可以找到红外接收模块，该模块专门用于接收红外发射模块发出的红外信号。
- 2、Arduino主控板上D12管脚已经集成了一个红外接收传感器。
- 3、本案例外接了一个红外传感器，连接在D5管脚上。
- 4、本案例配套了一个红外软硅胶遥控器。



知识接入



- 1、上传程序后，打开串口监视器，用遥控器对准红外接收传感器，按下任意按键，可以看到一个十六进制的编码。每个按键对应不同的编码，按住某个按键不放，会发送一个重复编码“FFFFFF”
- 2、十六进制由0-9，A-F共十六个字符组成，遵循“十六进一”规则。规定在十六进制的数值前面加上前缀0x。



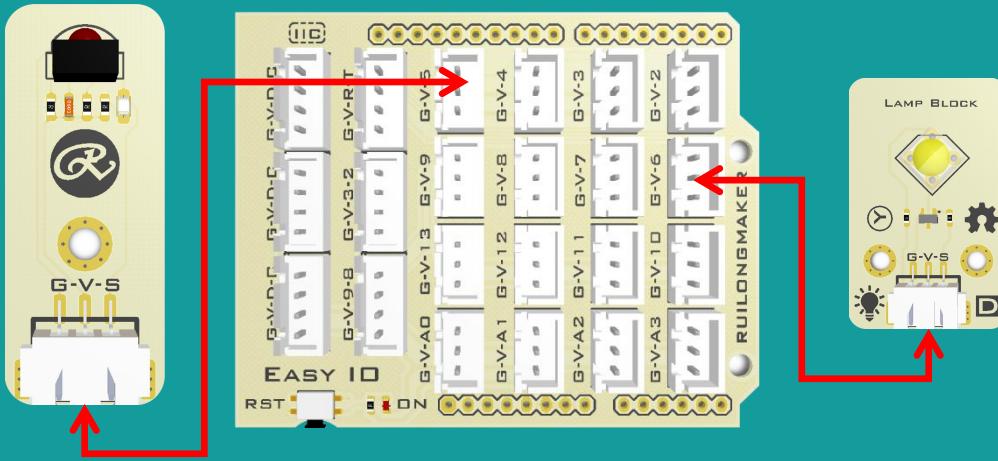
参考表格



键值	数值(H)	键值	数值(H)
A	FFA25D	0	FF6897
B	FF629D	1	FF30CF
C	FFE21D	2	FF18E7
D	FF22DD	3	FF7A85
E	FFC23D	4	FF10EF
F	FFB04F	5	FF38C7
上	FF02FD	6	FF5AA5
下	FF9867	7	FF42BD
左	FFE01F	8	FF4AB5
右	FF906F	9	FF52AD
中间R	FFA857		



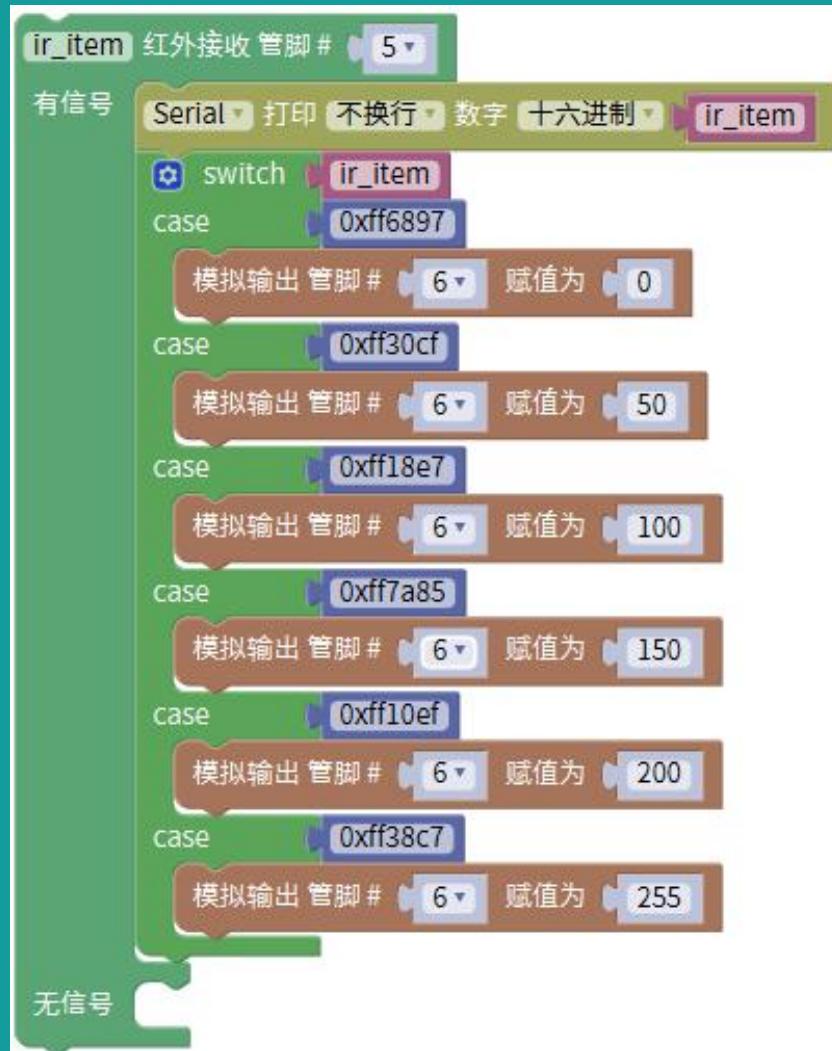
硬件连线



序号	名称	连线	备注
1	红外接收传感器	D5	
2	闪灯模块	D6	



程序编程



- 1、串口打印红外接收传感器接收到的编码值。
- 2、0xff6879-0xff38c7对应了红外遥控器的0-5按键。
- 3、总共划分了6个亮度级别，按键0代表不亮，按键5代表最亮。
- 4、判断it_item的值，实现对闪灯模块亮度的调节。



1、头文件引入

```
#include <IRremote.h>
```

2、定义变量

```
long ir_item;
```

3、创建对象

```
IRrecv irrecv_5(5); // 创建带有一个实参的对象
```

4、结构体变量

```
decode_results results_5; // 创建结构体变量results_5
```

5、setup函数

```
void setup(){
  Serial.begin(9600);
  irrecv_5.enableIRIn(); // 调用enableIRIn函数
}
```

6、loop函数

```
void loop(){
  if (irrecv_5.decode(&results_5)) { // 调用decode函数，参数为结构体地址
    ir_item=results_5.value; // ir_item赋值接收到的编码值
    String type="UNKNOWN";
    String typelist[18]={"UNUSED", "RC5", "RC6", "NEC", "SONY", "PANASONIC", "JVC", "SAMSUNG",
    "WHYNTER", "AIWA_RC_T501", "LG", "SANYO", "MITSUBISHI", "DISH", "SHARP", "DENON",
    "PRONTO", "LEGO_PF"};;
    if(results_5.decode_type>=1&&results_5.decode_type<=17){
      type=typelist[results_5.decode_type];
    }
    Serial.println("IR TYPE:"+type+" ");
    Serial.print(ir_item,HEX);
    switch (ir_item) {
      case 0xff6897:
        analogWrite(6,0);
        break;
      case 0xff30cf:
        analogWrite(6,50);
        break;
      case 0xff18e7:
        analogWrite(6,100);
        break;
      case 0xff7a85:
        analogWrite(6,150);
        break;
      case 0xff10ef:
        analogWrite(6,200);
        break;
      case 0xff38c7:
        analogWrite(6,255);
        break;
    }
    irrecv_5.resume();
  } else {
  }
}
```



项目简析



项目任务

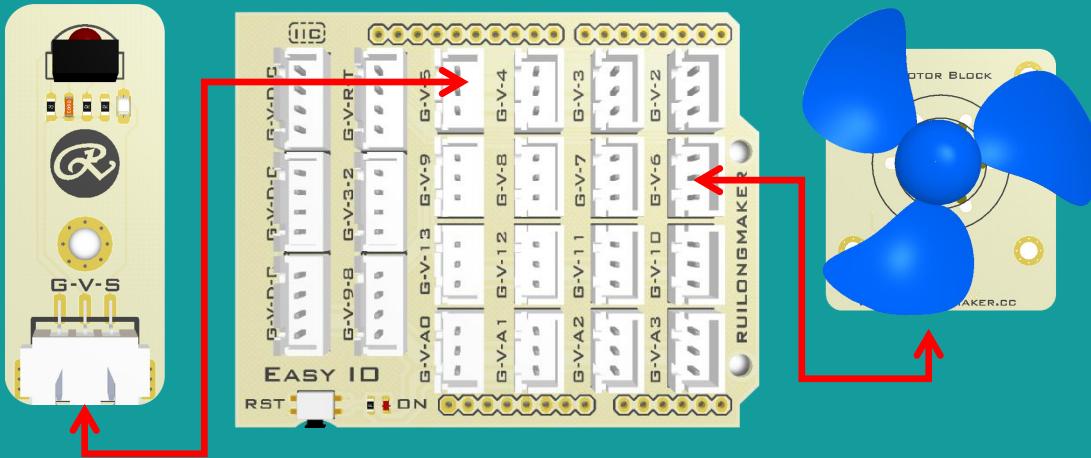
红外遥控被广泛的应用在家用电器和工业控制等，由发射和接收两部分组成，遥控器发出调制过的红外光波，接收器将遥控器发射的红外光转化为相应的电信号。利用红外接收传感器和马达风扇模块，制作一个遥控调速风扇，按下中间“R”按钮，对风扇开关进行控制，按下左按钮，风扇速度减小，按下右按钮，风扇速度增大。

思路分析

Arduino主控板已经集成了一个红外接收传感器，本套件中搭载了一个单独的红外接收传感器，可以用于外接。红外遥控器的每个按键对应了不同的编码，不同的遥控器编码也不相同，可以通过串口监视器打印不同按键对应的编码值，判断该编码值要对应执行的语句，就可以轻松的实现红外遥控。



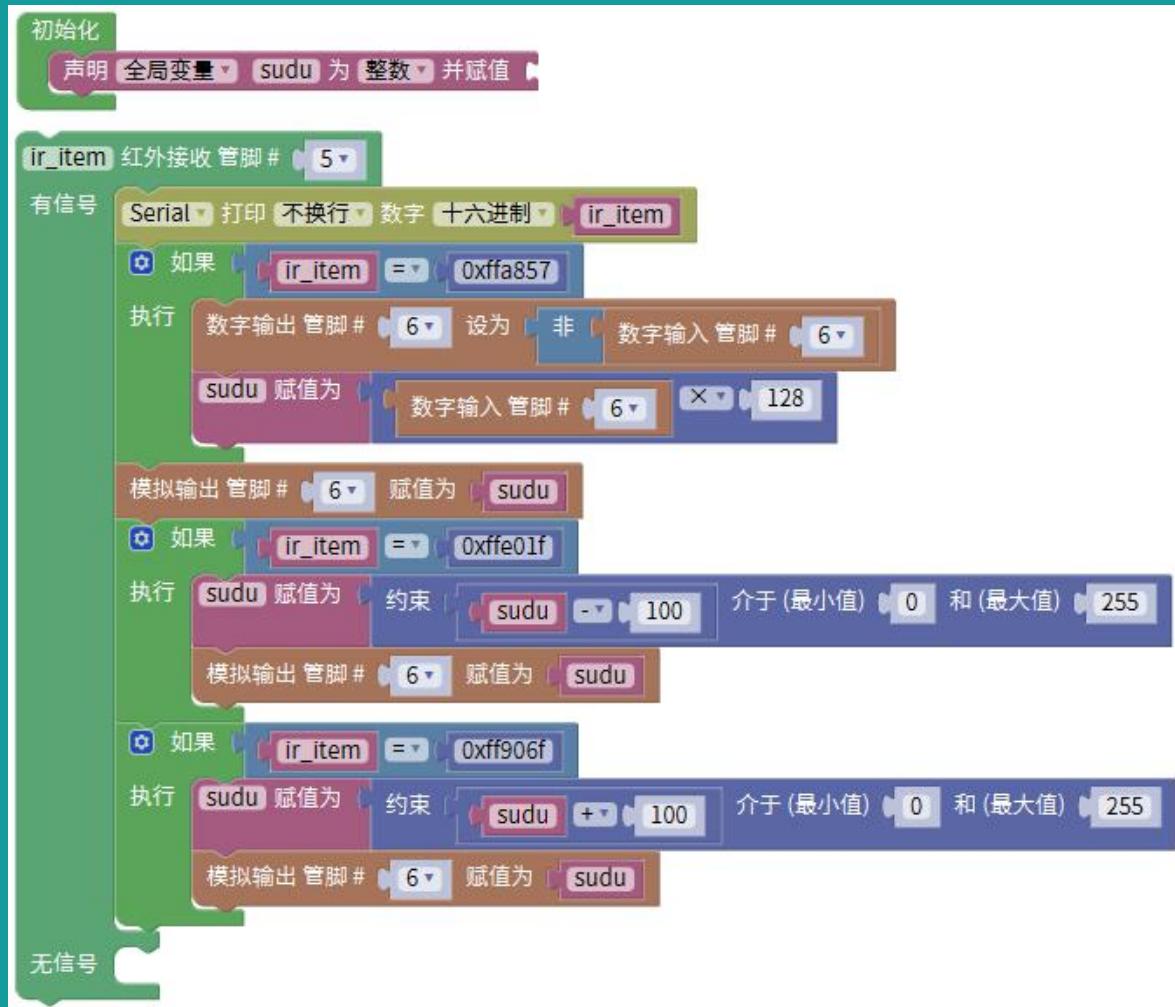
硬件连线



序号	名称	连线	备注
1	红外接收传感器	D5	
2	马达风扇模块	D6	



程序编程



- 1、声明sudu变量，用于保存风扇初始速度。
- 2、串口打印红外接收传感器接收编码。
- 3、如果遥控器中间“R”按键被按下，对6号管脚的连接的马达风扇模块的状态进行判断，如果默认为开，按下按钮，风扇关闭。风扇连接在模拟输出管脚，模拟值范围为0-255，风扇的初始速度赋值中间值。
- 4、如果按下左按钮，对风扇速度以100的幅度减小，并且约束模拟值处于0-255范围之内。
- 5、如果按下右按钮，对风扇速度以100幅度增大，并且约束模拟值处于0-255范围。



1、头文件引入

```
#include <IRremote.h>
```

2、定义变量

```
volatile int sudu;  
long ir_item;
```

3、创建对象

```
IRrecv irrecv_5(5);
```

4、结构体变量

```
decode_results results_5;//创建结构体变量results_5
```

5、setup函数

```
void setup(){  
    sudu = 0;  
    Serial.begin(9600);  
    pinMode(6, OUTPUT);  
    irrecv_5.enableIRIn();  
}
```

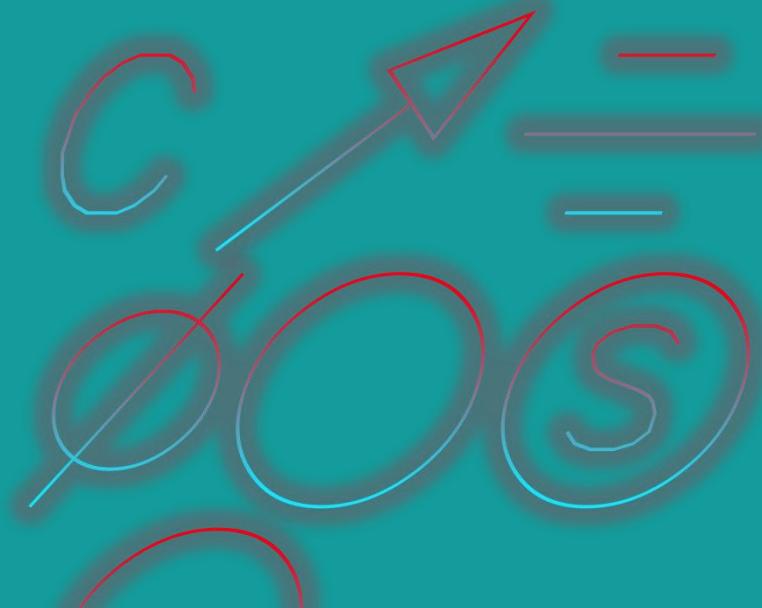
6、loop函数

```
void loop(){  
    if (irrecv_5.decode(&results_5)) {  
        ir_item=results_5.value;  
        String type="UNKNOWN";  
        String typelist[18]={"UNUSED", "RC5", "RC6", "NEC", "SONY", "PANASONIC", "JVC", "SAMSUNG",  
"WHYNTER", "AIWA_RC_T501", "LG", "SANYO", "MITSUBISHI", "DISH", "SHARP", "DENON",  
"PRONTO", "LEGO_PF"};  
        if(results_5.decode_type>=1&&results_5.decode_type<=17){  
            type=typelist[results_5.decode_type];  
        }  
        Serial.println("IR TYPE:"+type+" ");  
        Serial.print(ir_item,HEX);  
        if (ir_item == 0xffa857) {  
            digitalWrite(6,!digitalRead(6));  
            sudu = digitalRead(6) * 128;  
        }  
        analogWrite(6,sudu);  
        if (ir_item == 0xffe01f) {  
            sudu = constrain(sudu - 100, 0, 255);  
            analogWrite(6,sudu);  
        }  
        if (ir_item == 0xff906f) {  
            sudu = constrain(sudu + 100, 0, 255);  
            analogWrite(6,sudu); }  
        irrecv_5.resume();  
    } else {  
    }  
}
```



项目简析

English



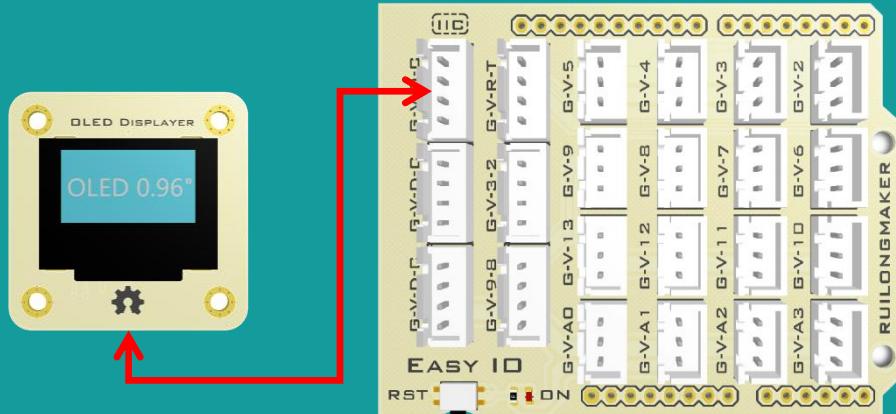
项目任务 利用SSD1306显示屏，显示中英文文字。

思路分析

初始化SSD1306显示屏，显示屏的大小为128x64,横坐标范围为[0,127],纵坐标范围为[0,63],显示英文字体确定起点坐标，可直接显示英文字体，中文字体的显示可以通过图像显示，保存到flash中。



硬件连线



序号	名称	连线	备注
1	OLED显示器	GVDC	

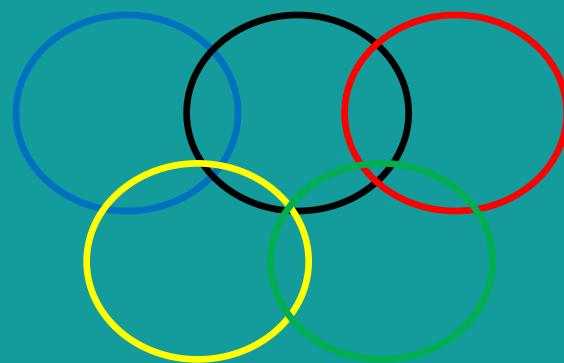


程序编程





项目简析



项目任务

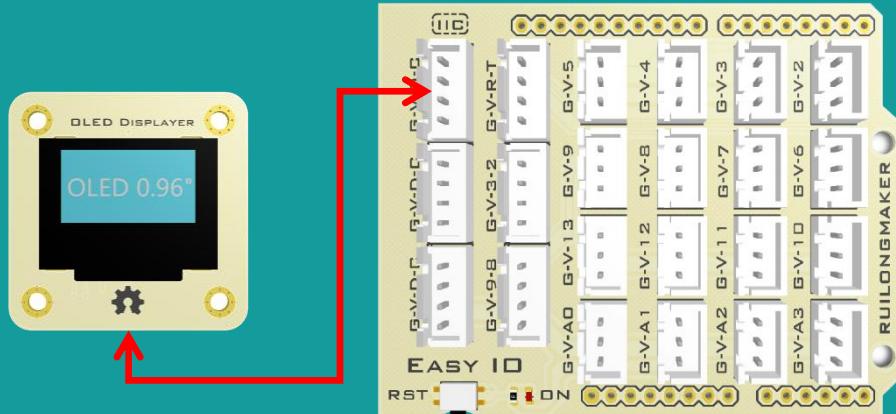
利用SSD1306显示屏，显示奥运五环图案

思路分析

初始化SSD1306显示屏，显示屏的大小为128x64,横坐标范围为[0,127],纵坐标范围为[0,63],屏幕左上角为[0,0],确定五个圆圆心在坐标系中的位置，即可显示出奥运五环图案。



硬件连线



序号	名称	连线	备注
1	OLED显示器	GVDC	



程序编程





1、头文件引入

```
#include <U8g2lib.h>
#include <Wire.h>
```

2、创建对象

```
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);
```

3、显示页面函数

```
void page1() {
    u8g2.drawRFrame(1,1,124,62,10); //显示圆角矩形参数分别为x坐标, y坐标, 宽度、高度、圆角半径
    u8g2.drawCircle(64,32,10,U8G2_DRAW_ALL); //显示圆形, 参数为圆形横纵坐标, 半径, 整圆
    u8g2.drawCircle(38,32,10,U8G2_DRAW_ALL);
    u8g2.drawCircle(90,32,10,U8G2_DRAW_ALL);
    u8g2.drawCircle(51,42,10,U8G2_DRAW_ALL);
    u8g2.drawCircle(77,42,10,U8G2_DRAW_ALL);
}
```

4、setup函数

```
void setup(){
    u8g2.setI2CAddress(0x3C*2);
    u8g2.begin();
}
```

5、loop函数

```
void loop(){
    u8g2.firstPage(); //刷新页面
    do
    {
        page1(); //调用显示函数
    }while(u8g2.nextPage());
}
```



项目简析



项目任务

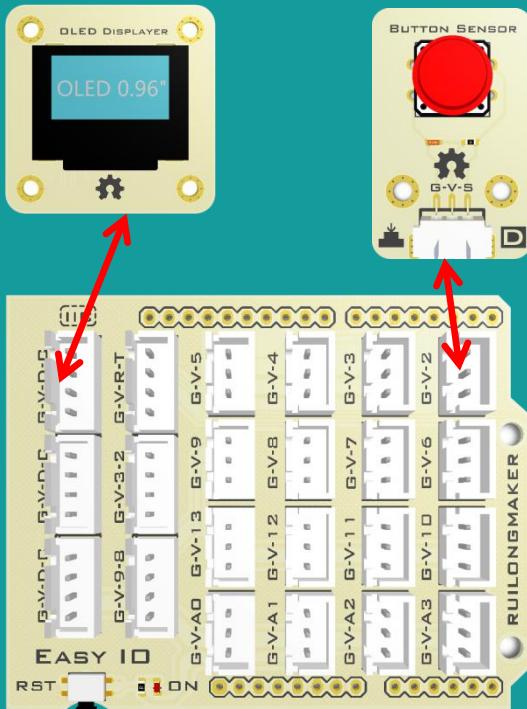
利用SSD1306显示屏，制作一个小游戏真心话大冒险。

思路分析

初始化一个布尔变量、一个整型变量，SSD1306,编写真心话和大冒险函数，布尔型变量用于判断执行真心话函数还是大冒险函数，整型变量切换问题的显示，执行哪个page页面。



硬件连线



序号	名称	连线	备注
1	OLED显示器	GVDC	
2	按钮传感器	D2	



程序编程

初始化

```
初始化 SSD1306(128×64) u8g2 屏幕 旋转 0° SCL SCL SDA SDA 设备地址 0x3C
声明 全局变量 random 为 布尔 并赋值 0
声明 全局变量 command 为 整数 并赋值 0
```

page1

```
执行 显示屏 u8g2 设置英文字体 Times New Roman 字号 08 字形 常规
显示屏 u8g2 显示文本 起点x 0 起点y 40 内容 "truth:dislike people"
```

page2

```
执行 显示屏 u8g2 设置英文字体 Times New Roman 字号 08 字形 常规
显示屏 u8g2 显示文本 起点x 0 起点y 40 内容 "truth:your idol"
```

page3

```
执行 显示屏 u8g2 设置英文字体 Times New Roman 字号 08 字形 常规
显示屏 u8g2 显示文本 起点x 0 起点y 40 内容 "truth:of the person you like"
```

page4

```
执行 显示屏 u8g2 设置英文字体 Times New Roman 字号 08 字形 常规
显示屏 u8g2 显示文本 起点x 0 起点y 40 内容 "dare:sing a song"
```

page5

```
执行 显示屏 u8g2 设置英文字体 Times New Roman 字号 08 字形 常规
显示屏 u8g2 显示文本 起点x 0 起点y 40 内容 "dare:dance"
```

page6

```
执行 显示屏 u8g2 设置英文字体 Times New Roman 字号 08 字形 常规
显示屏 u8g2 显示文本 起点x 0 起点y 40 内容 "dare:jump 20 times"
```

truth 参数: command

```
执行 如果 command = 1
执行 显示屏 u8g2 刷新页面
执行 page1
否则如果 command = 2
执行 显示屏 u8g2 刷新页面
执行 page2
否则如果 command = 3
执行 显示屏 u8g2 刷新页面
执行 page3
```

dare 参数: command

```
执行 如果 command = 1
执行 显示屏 u8g2 刷新页面
执行 page4
否则如果 command = 2
执行 显示屏 u8g2 刷新页面
执行 page5
否则如果 command = 3
执行 显示屏 u8g2 刷新页面
执行 page6
```

如果 非 数字输入管脚 # 2
执行 延时 毫秒 10
如果 非 数字输入管脚 # 2
执行 random 赋值为 随机整数从 0 到 2
如果 random
执行 truth 参数: command
随机整数从 1 到 4
否则
执行 dare 参数: command
随机整数从 1 到 4
重复 满足条件 非 数字输入管脚 # 2
执行



1、头文件引入

```
#include <U8g2lib.h>
#include <Wire.h>
```

2、定义变量

```
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);
volatile boolean random2;
volatile int command;
```

3、真心话函数

```
void truth(int command) {
    if (command == 1) { // 判断随机数的值
        u8g2.firstPage(); // 刷新页面
        do
        {
            page1();
        }while(u8g2.nextPage());
    } else if (command == 2) {
        u8g2.firstPage();
        do
        {
            page2();
        }while(u8g2.nextPage());
    } else if (command == 3) {
        u8g2.firstPage();
        do
        {
            page3();
        }while(u8g2.nextPage());
    }
}
```

4、大冒险函数

```
void dare(int command) {
    if (command == 1) {
        u8g2.firstPage();
        do
        {
            page4();
        }while(u8g2.nextPage());
    } else if (command == 2) {
        u8g2.firstPage();
        do
        {
            page5();
        }while(u8g2.nextPage());
    } else if (command == 3) {
        u8g2.firstPage();
        do
        {
            page6();
        }while(u8g2.nextPage());
    }
}
```

5、显示页面函数

```
void page1() {
    u8g2.setFont(u8g2_font_timR08_tf); // 设置显示字体
    u8g2.setFontPosTop();
    u8g2.setCursor(0,40); // 设置显示位置
    u8g2.print("truth:dislike people"); // 打印真心话，讨厌的人
}
```

```
void page2() {
    u8g2.setFont(u8g2_font_timR08_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(0,40);
    u8g2.print("truth:your idol");
}
```



```
void page3() {  
    u8g2.setFont(u8g2_font_timR08_tf);  
    u8g2.setFontPosTop();  
    u8g2.setCursor(0,40);  
    u8g2.print("truth:of the person you like");  
}  
  
void page4() {  
    u8g2.setFont(u8g2_font_timR08_tf);  
    u8g2.setFontPosTop();  
    u8g2.setCursor(0,40);  
    u8g2.print("dare:sing a song");  
}
```

```
void page5() {  
    u8g2.setFont(u8g2_font_timR08_tf);  
    u8g2.setFontPosTop();  
    u8g2.setCursor(0,40);  
    u8g2.print("dare:dance");  
}  
  
void page6() {  
    u8g2.setFont(u8g2_font_timR08_tf);  
    u8g2.setFontPosTop();  
    u8g2.setCursor(0,40);  
    u8g2.print("dare:jump 20 times");  
}
```

6、setup函数

```
void setup(){  
    u8g2.setI2CAddress(0x3C*2);//初始化SSD显示屏  
    u8g2.begin();  
    random2 = 0;  
    command = 0;  
    u8g2.enableUTF8Print();  
    pinMode(2, INPUT);//2号管脚设为输入模式  
}
```

6、loop函数

```
void loop(){  
    if (!digitalRead(2)) {//按钮传感器默认为1，按下为0，按下按钮对电平进行取反  
        delay(10); //延时  
        if (!digitalRead(2)) {  
            random2 = random(0, 2); //取随机数范围，值为0或者1  
            if (random2) {  
                truth(random(1, 4)); //执行真心话函数  
            } else {  
                dare(random(1, 4)); //执行大冒险函数  
            }  
        }  
        while (!digitalRead(2)) {  
        }  
    }  
}
```



Arduino 片上应用



项目简析



项目任务

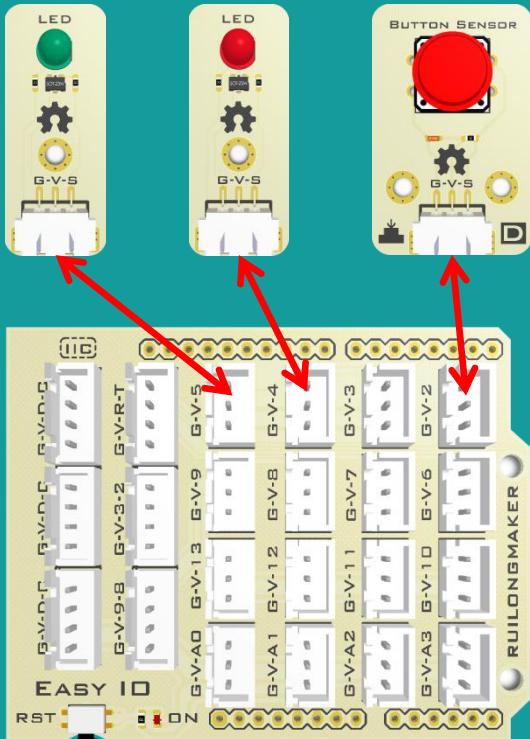
使用一个按钮传感器和两个LED灯来模拟双闪灯的控制，按下按钮时，LED灯交替亮灭，再次按下按钮，灯灭，且两个灯状态保持同步。

思路分析

初始化两个整型变量item, item1, 一个时间变量time用长整型来保存，按下按钮传感器时，item赋值为1-item，可以实现按钮传感器对双闪的开关的控制，获取系统运行时间，如果时间间隔大于1000ms,item1赋值为1-item1, item1的取值范围为0, 1，当item和item1都为1的时候，灯亮。重新对时间变量赋值为系统运行时间。



硬件连线



序号	名称	连线	备注
1	按钮传感器	D2	
2	LED模块	D4	
3	LED模块	D5	



程序编程



时间用long

按键消抖

按下按钮，
item0、1之间切换，控制双闪灯开关

切换亮灭后
time重新赋值

```
volatile int item;//定义变量
volatile int item1;
volatile long time;
void setup(){
    item = 0;
    item1 = 0;
    time = 0;
    pinMode(2, INPUT); //管脚2设置为输入模式
    pinMode(4, OUTPUT); //管脚4设置为输出模式
    pinMode(5, OUTPUT);
}
void loop(){
    if (!digitalRead(2)) {
        while (!digitalRead(2)) {
            delay(10); //按键消抖
        }
        item = 1 - item; //状态变量 取值为0或1
    }
    if (millis() - time >= 1000) { //millis获取系统运行时间
        item1 = 1 - item1;
        digitalWrite(4,(item * item1)); //管脚4写入0或1，代表亮或者不亮
        digitalWrite(5,(item * item1));
        time = millis();
    }
}
```



项目简析



项目任务

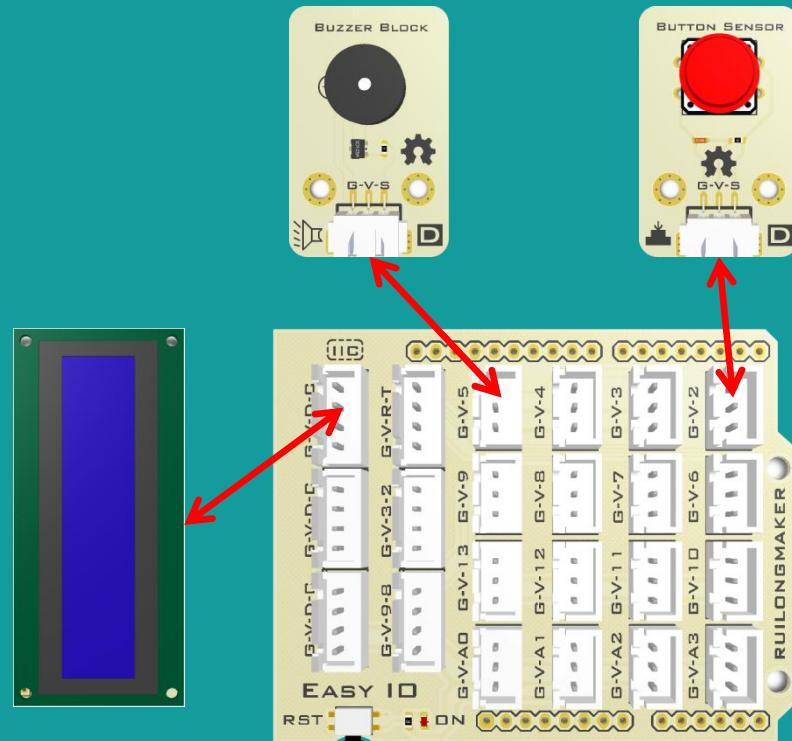
使用一个按钮传感器、LCD显示屏、蜂鸣器模块实现单击按钮，设置倒计时时间，度量为分钟，双击按钮开始倒计时，倒计时结束后，蜂鸣器蜂鸣十次。

思路分析

Arduino可以实现按键的单双击识别，当单击按下按钮时，设置倒计时时间，度量为分钟，即按下次，时间增长一分钟，对设置的时间除以10实现对十位的分离，%10实现对个位的分离，分离完成后显示在LCD显示屏上，双击按钮，开始倒计时，利用循环，延时1s可实现倒计时，倒计时结束后，蜂鸣器蜂鸣十次。



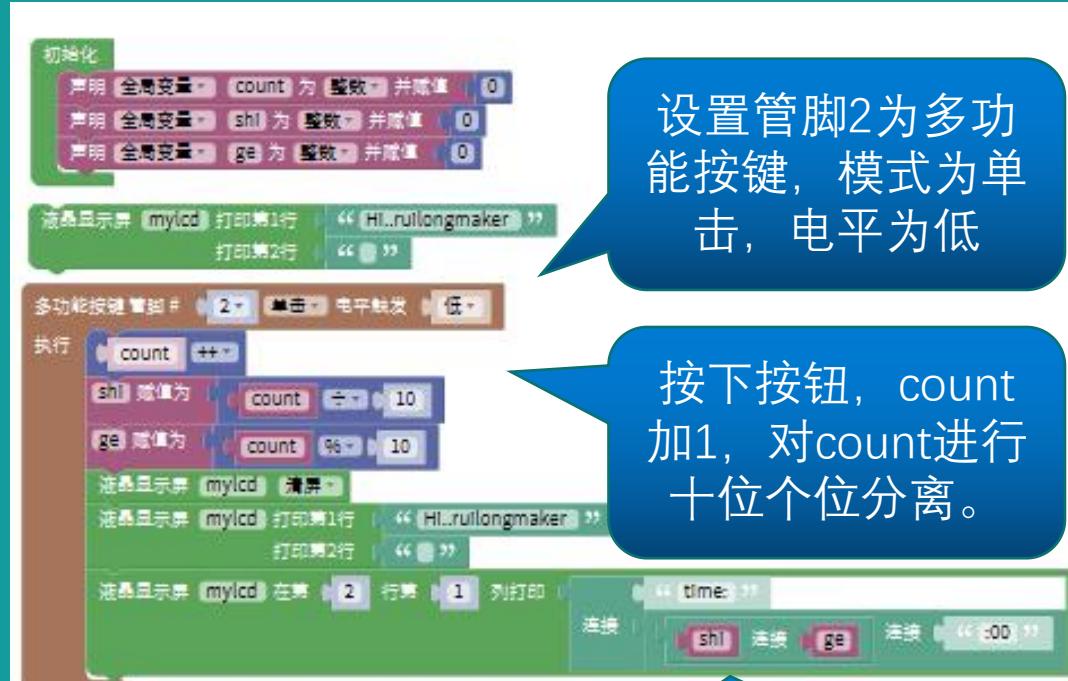
硬件连线



序号	名称	连线	备注
1	LCD显示器	GVDC	
2	按键传感器	D2	
3	蜂鸣器模块	D5	



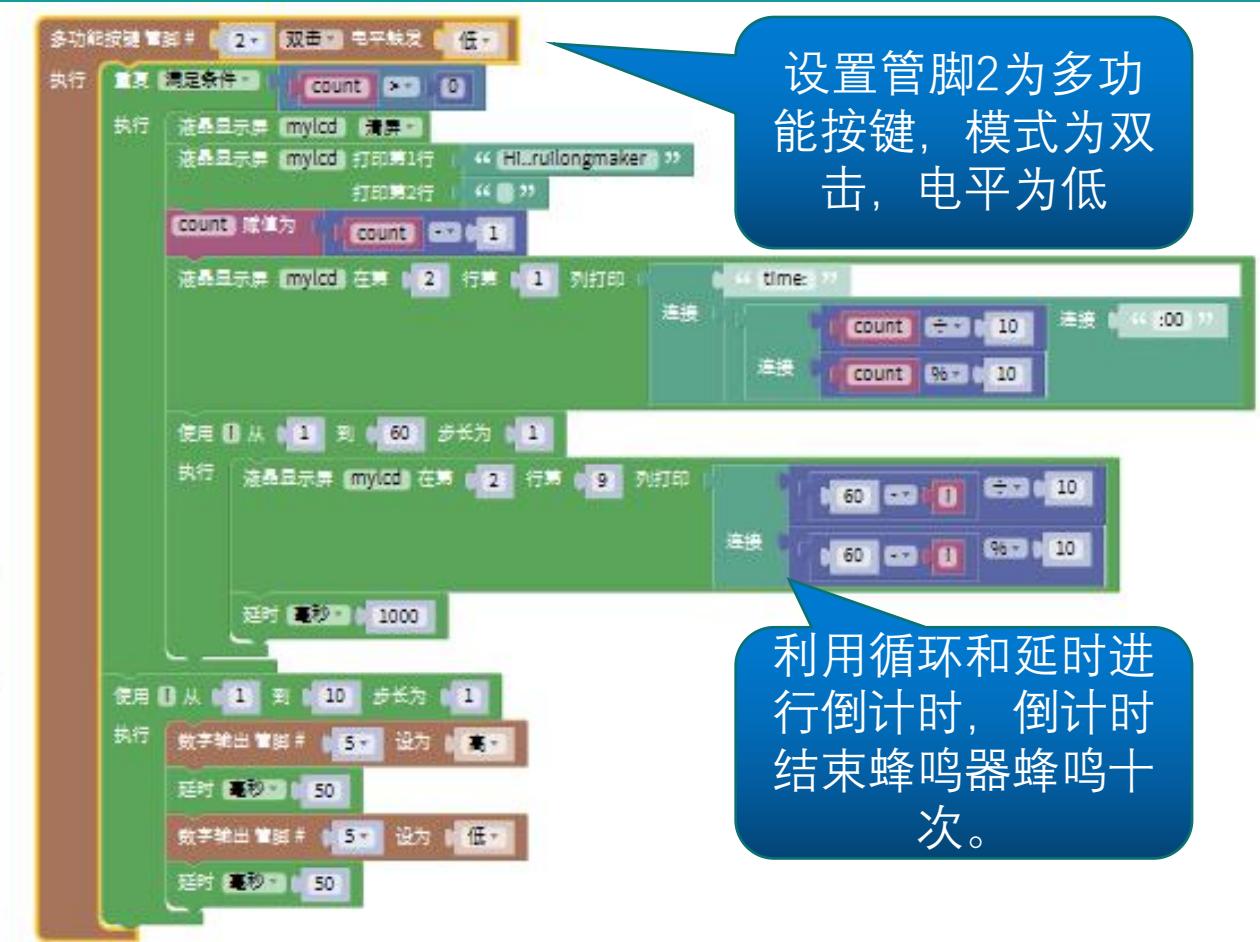
程序编程



设置管脚2为多功能按键，模式为单击，电平为低

按下按钮，count加1，对count进行十位个位分离。

对分离好的数据按照逻辑显示在LCD显示屏上



设置管脚2为多功能按键，模式为双击，电平为低

利用循环和延时进行倒计时，倒计时结束蜂鸣器蜂鸣十次。



1、头文件引入

```
#include <OneButton.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

2、定义变量

```
volatile int count;
volatile int shi;
volatile int ge;
```

3、定义对象

```
OneButton button2(2,true);
LiquidCrystal_I2C mylcd(0x27,16,2);
```

4、按键双击函数

```
void attachDoubleClick2() {
    while (count > 0) {
        mylcd.clear();//清除LCD显示屏
        mylcd.setCursor(0, 0);//设置打印开始位置
        mylcd.print("Hi..ruilongmaker");//第一行打印
        mylcd.setCursor(0, 1);
        mylcd.print("");
        count = count - 1;
        mylcd.setCursor(1-1, 2-1);
        mylcd.print(String("time:") + String(String((count / 10)) + String(((long) (count) % (long)
(10)))) + String(":00"));//显示开始倒计时时间
        for (int i = 1; i <= 60; i = i + (1)) {
            mylcd.setCursor(9-1, 2-1);
            mylcd.print(String(((60 - i) / 10)) + String(((long) ((60 - i) % (long) (10))));//显示倒计时
            delay(1000); }
        for (int i = 1; i <= 10; i = i + (1)) {
            digitalWrite(5,HIGH);//5号管脚写入高电平
            delay(50);
            digitalWrite(5,LOW);//5号管脚写入低电平
            delay(50);
        }
    }
}
```

5、按键单击函数

```
void attachClick2() {
    count++;//倒计时时间
    shi = count / 10;//分离十位
    ge = (long) (count) % (long) (10);//分离个位
    mylcd.clear();
    mylcd.setCursor(0, 0);
    mylcd.print("Hi..ruilongmaker");
    mylcd.setCursor(0, 1);
    mylcd.print("");
    mylcd.setCursor(1-1, 2-1);
    mylcd.print(String("time:") + String(String(shi) + String(ge)) + String(":00"));//显示倒计时时间
}
```

6、setup函数

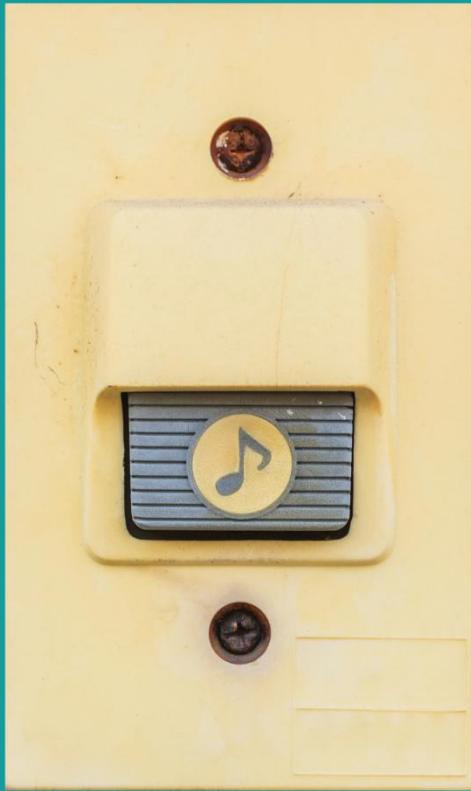
```
void setup(){
    count = 0;
    shi = 0;
    ge = 0;
    button2.attachDoubleClick(attachDoubleClick2);
    mylcd.init();
    mylcd.backlight();
    pinMode(5, OUTPUT);
    button2.attachClick(attachClick2);
}
```

7、loop函数

```
void loop(){
    button2.tick();
    mylcd.setCursor(0, 0);
    mylcd.print("Hi..ruilongmaker");
    mylcd.setCursor(0, 1);
    mylcd.print("");
    button2.tick();
}
```



项目简析



项目任务

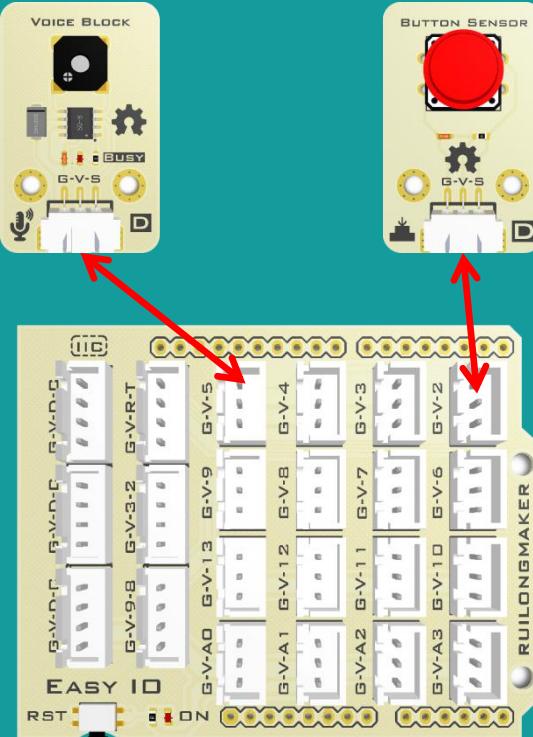
“中断”是指在某些情况下，主控板要暂停执行当前的程序而执行新的程序。Arduino能够接收中断信号的管脚只有两个D2和D3管脚，中断被触发的条件为：上升（低电平变为高电平），下降（高电平变为低电平），改变（电平改变），本案例所用按钮传感器默认为高，按下为低，触发中断模式为下降。使用按钮传感器和语音模块实现按下按钮，播放声音。

思路分析

初始化布尔型变量用于判断是否播放音乐，初始化赋值为假，设置软件中断，模式为下降，按下按钮，发生中断，item赋值为非item，播放声音；在播放声音时按下按钮，产生中断，停止播放声音。播放结束后item赋值为假。



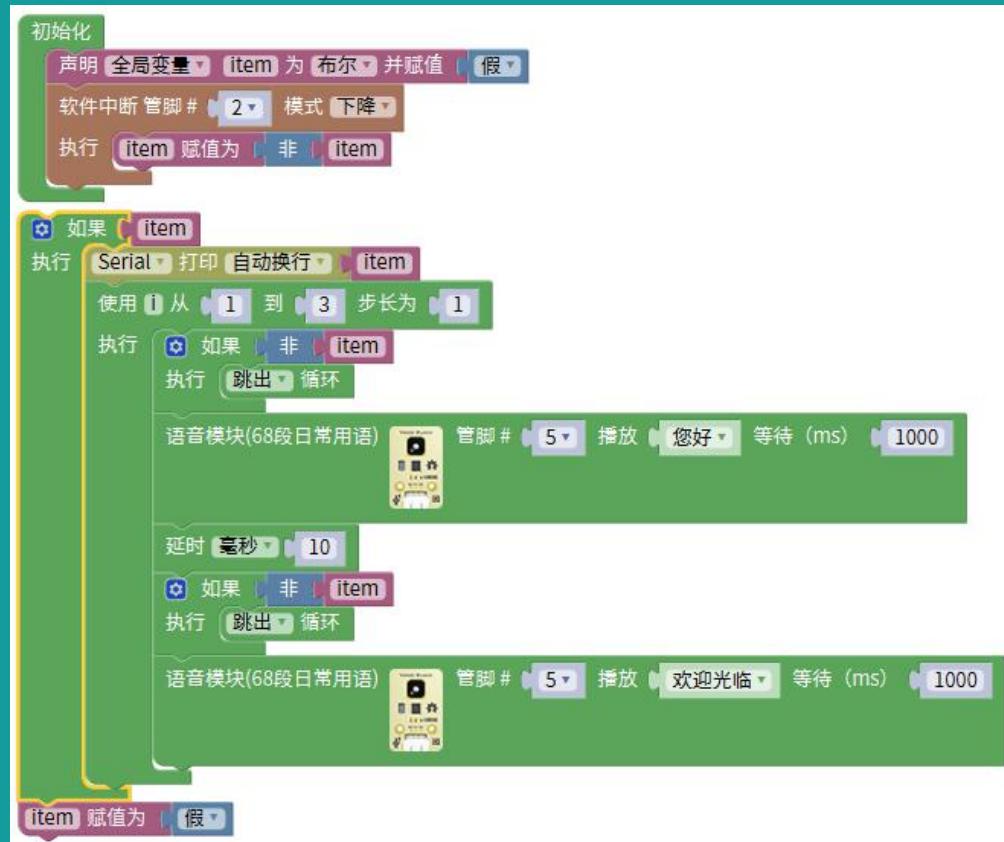
硬件连线



序号	名称	连线	备注
1	按钮传感器	D2	
2	声音传感器	D5	



程序编程



- 1、初始化设置布尔型item变量，赋值为假，设置中断监听，当发生中断时，item切换到非item。
- 2、按下按钮，item为真。
- 3、连续三次播放“您好，欢迎光临”。
- 4、在播放声音时按下按钮，即产生中断，停止播放声音。
- 5、播放声音中途没有按下按钮，连续三次播放结束后，item赋值为假，停止播放。



1、头文件引入

```
#include<RL_Voice68.h>
#include <PinChangelnt.h>
```

2、定义变量

```
volatile boolean item;
```

3、定义对象

```
VOICE_68 voice_5(5);
```

4、软件中断函数

```
void attachPinInterrupt_fun_2() {
    item = !item;}
```

5、setup函数

```
void setup(){
    item = false;
    pinMode(2, INPUT);//管脚2设置为输入模式
    PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,FALLING);//类外函数
    Serial.begin(9600);//串口比特率
    voice_5.begin();
}
```

6、loop函数

```
void loop(){
if (item) {
    Serial.println(item);
    for (int i = 1; i <= 3; i = i + (1)) {
        if (!item) {
            break;//如果! item为假, 跳出循环
        }
        voice_5.send_data(0x14); //volume control 0xE0-E7;//播放“您好”指令
        delay(1000);
        if (!item) {
            break;
        }
        voice_5.send_data(0x19); //volume control 0xE0-E7;//播放“欢迎光临”指令。
        delay(1000);}}
    item = false;}
```



项目简析



项目任务

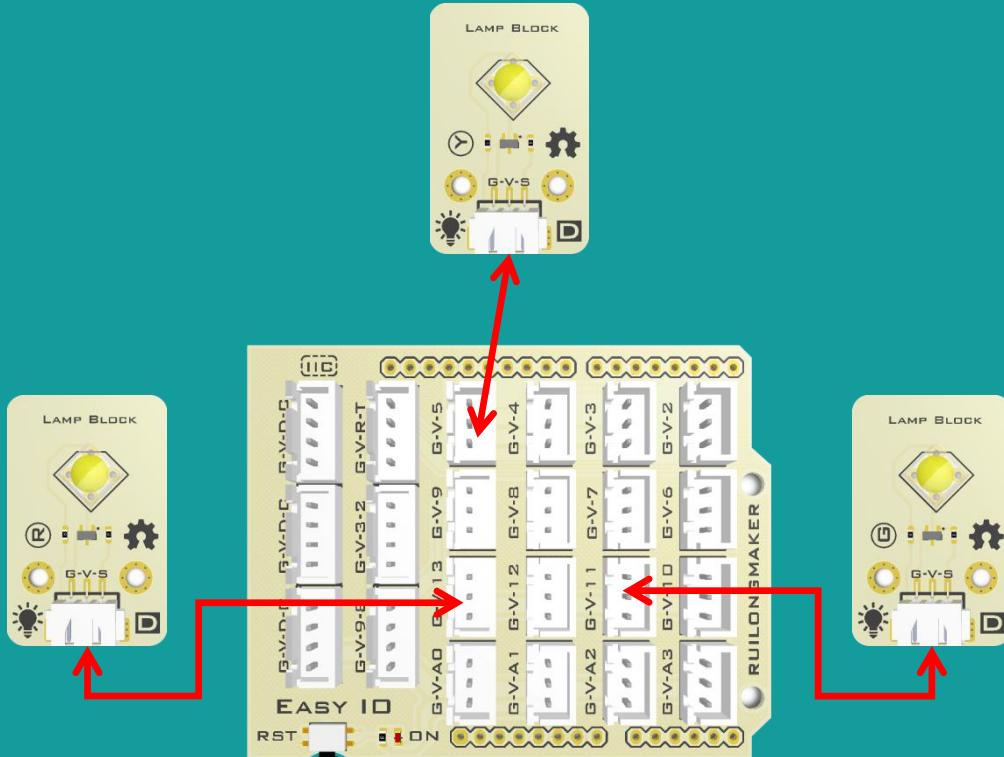
“中断”是指在某些情况下，主控板要暂停执行当前的程序而执行新的程序。中断有很多类型，Arduion自带的定时器可让定时中断发生，可以精确的控制时间，使用定时器中断实现闪光灯的效果。

思路分析

使用简单定时器，可直接设置时间间隔和定时器，使用MsTimer2，需要放在初始化模块中，只能有一个，配合启动语句一起使用。



硬件连线



序号	名称	连线	备注
1	闪灯模块（黄色）	D5	
2	闪灯模块（绿色）	D11	
3	闪灯模块（红色）	D13	



程序编程



- 1、定时器MsTimer2放在初始化中，程序中只能有一个。
- 2、简单定时器最多可以设置16个
- 3、定时器MsTimer2要配合启动模块一起使用。



1、头文件引入

```
#include <MsTimer2.h>
```

2、定义变量

```
volatile int item;
```

3、定时器函数

```
void msTimer2_func() {
    item++; // 变量自增
    digitalWrite(5, (!digitalRead(5))); // 给5号管脚写入读取到的5号管脚的电平值取反，实现亮灭的切换。
    if (item == 1) {
        digitalWrite(11, (!digitalRead(11))); // 亮灭切换
    }
    if (item == 2) {
        digitalWrite(13, (!digitalRead(13))); // 亮灭切换
        item = 0;
    }
}
```

4、setup函数

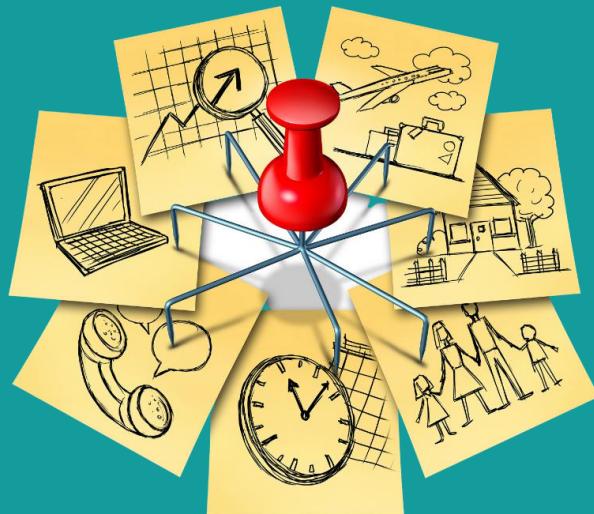
```
void setup(){
    item = 0;
    pinMode(5, OUTPUT); // 5号管脚设置为输出模式
    pinMode(11, OUTPUT); // 11号管脚设置为输出模式
    pinMode(13, OUTPUT);
    MsTimer2::set(1000, msTimer2_func); // MsTimer2类外函数，参数值1000，代表时间，msTimer2_func是一个函数，代表每隔1000ms，执行一次 msTimer2_func函数。
    MsTimer2::start();
}
```

5、loop函数

```
void loop(){
}
```



项目简析



项目任务

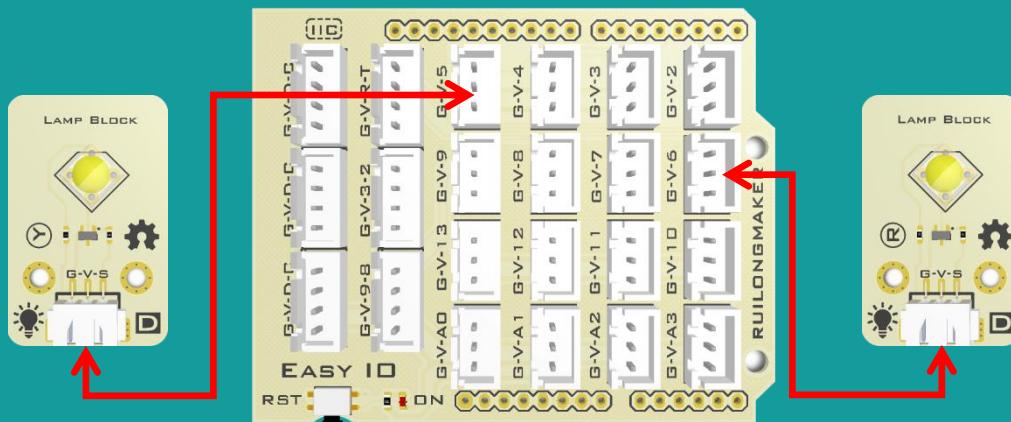
多线程是指从软件或者硬件上实现多个线程并发执行的技术，Arduino开发板借助SCoop即可轻松实现多线程。利用闪灯模块进行多线程的实验。

思路分析

Arduino多线程的实现步骤为：首先引入SCoop库，设置setup、设置loop，定义并创建线程实现具体功能。需要注意的是延迟函数使用sleep()，不要使用delay()；sleep()只在当前线程进行延迟，delay()则会在全局进行延迟。



硬件连线



序号	名称	连线	备注
1	闪灯模块（黄色）	D5	
2	闪灯模块（红色）	D6	



程序编程



- 1、Arduino可借助SCoop可以轻松实现多线程。
- 2、实现步骤为引入SCoop库、设置setup、设置loop、定义线程并实现功能。
- 3、在某个线程中想要暂停，使用sleep ()，如果使用delay () 整个程序都会暂停。



1、头文件引入

```
#include "SCoop.h"
```

2、创建线程scoopTask1

```
defineTask(scoopTask1)
```

3、类外函数

```
void scoopTask1::setup()//设定线程
{
    digitalWrite(5,LOW);//向5号管脚写入低电平
}
void scoopTask1::loop()//设定线程循环
{
    digitalWrite(5,(!digitalRead(5)));//向5号管脚写入读取到的5号管脚电平相反值
    sleep(100);//延时100ms
}
defineTask(scoopTask2)//创建线程scoopTask2
void scoopTask2::setup(){
    analogWrite(6,0);//向6号管脚写入模拟值0
}
void scoopTask2::loop()
{
    for (int i = 0; i <= 255; i = i + (1)) {
        analogWrite(6,i);
        sleep(5);}
    for (int i = 255; i >= 0; i = i + (-1)) {
        analogWrite(6,i);
        sleep(5);}}
```

5、setup函数

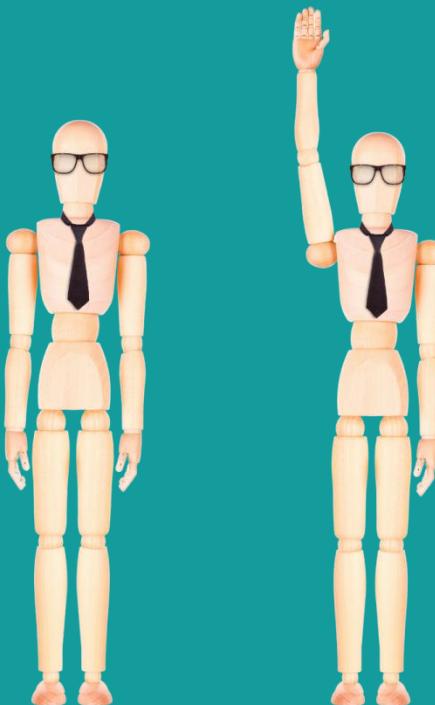
```
void setup()//设置setup
pinMode(5, OUTPUT);//5号管脚设置为输出模式
mySCoop.start();
}
```

loop函数

```
void loop(){
yield();}//执行线程
```



项目简析



项目任务

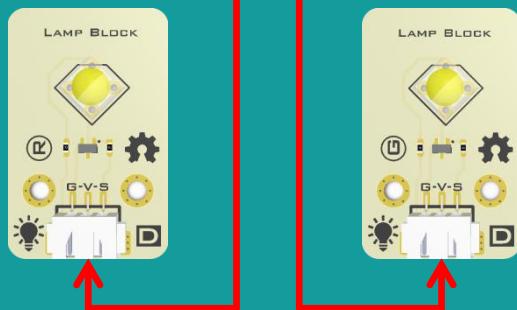
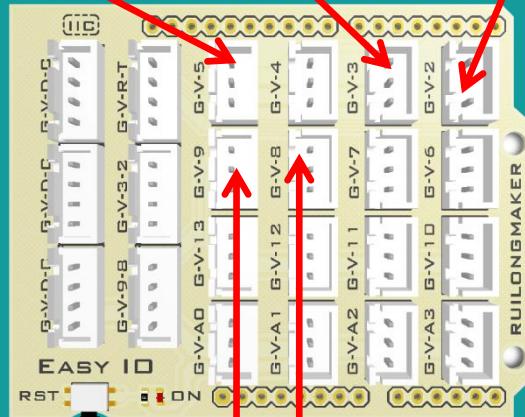
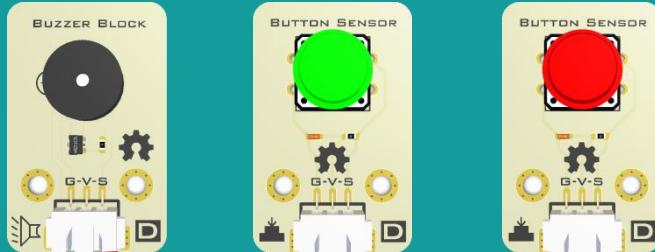
通过D2, D3管脚所连按钮触发中断，输入信号的互锁变化，控制蜂鸣器蜂鸣，相应闪灯模块亮起。

思路分析

将D2,D3管脚所连的按钮传感器设置为中断，声明两个标志位变量，当D2触发时，判断第二个标志位是否为真，如果为真，第一个标志位赋值为0，D3触发时，判断第一个标志位是否为真，如果为真，第二个标志位赋值为0，这时信号的互锁，根据标志位去判断相应要执行的代码。



硬件连线



序号	名称	连线	备注
1	按钮传感器 (红色)	D2	
2	按钮传感器 (绿色)	D3	
3	蜂鸣器模块	D5	
4	闪灯模块 (绿色)	D8	
5	闪灯模块 (红色)	D9	



程序编程



信号互锁

- 1、声明标志位变量。
- 2、设置软件中断管脚D2,D3，按钮传感器默认抬起是高电平，按下是低电平，中断模式选择下降。
- 3、信号互锁
- 4、当D2触发时，红色闪灯模块亮，蜂鸣器蜂鸣。如果D2,D3都为抬起状态，灯灭，蜂鸣器停止蜂鸣。
- 5、当D3触发时，绿色闪灯模块亮，蜂鸣器蜂鸣，如果D2,D3都为抬起状态，灯灭，蜂鸣器停止蜂鸣。



1、头文件引入

```
#include <PinChangeInt.h>
```

2、定义变量

```
volatile int flag1;  
volatile int flag2;
```

3、软件中断函数

```
//管脚D2设置中断
```

```
void attachPinInterrupt_fun_2(){  
//信号互锁  
if (flag2) {  
    flag1 = 0;  
} else {  
    flag1 = 1;  
}  
}
```

```
//管脚D3设置中断
```

```
void attachPinInterrupt_fun_3(){  
if (flag1) {  
    flag2 = 0;  
} else {  
    flag2 = 1;  
}  
}
```

4、setup函数

```
void setup(){  
flag1 = 0;  
flag2 = 0;  
//管脚2设置为输入模式, 高阻抗  
pinMode(2, INPUT);  
//管脚3设置为输入模式, 高阻抗  
pinMode(3, INPUT);  
//在类外定义成员函数attachInterrupt,参数管脚号2, 中断, 模式为FALLING下降  
PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,FALLING);  
//在类外定义成员函数attachInterrupt,参数管脚号3, 中断, 模式为FALLING下降  
PCintPort::attachInterrupt(3,attachPinInterrupt_fun_3,FALLING);  
//管脚9设置为输出模式, 低阻态, 可以点亮LED。  
pinMode(9, OUTPUT);  
pinMode(5, OUTPUT);  
pinMode(8, OUTPUT);  
}
```

5、loop函数

```
void loop(){  
if (flag1) {  
    digitalWrite(9,HIGH);//9号管脚输出高电平  
    digitalWrite(5,HIGH);  
    if (digitalRead(2) && digitalRead(3)) {//按钮2和3抬起  
        digitalWrite(9,LOW);//9号管脚输出低电平  
        digitalWrite(5,LOW);  
        flag1 = 0;//标志位清零  
    }  
} else if (flag2) {  
    digitalWrite(8,HIGH);  
    digitalWrite(5,HIGH);  
    if (digitalRead(2) && digitalRead(3)) {  
        digitalWrite(8,LOW);  
        digitalWrite(5,LOW);  
        flag2 = 0; }}}
```



项目简析

EEPROM

项目任务

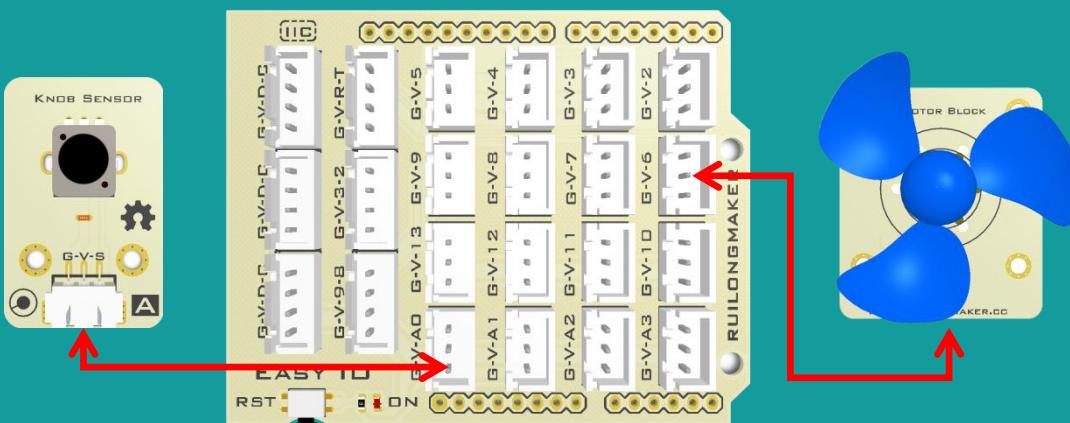
EEPROM指的是带电可擦可编程只读存储器，即通过编程改变EEPROM的值，即使断电数据也不会丢失。Arduino上EEPROM只需要两个函数就可以完成对数据的写与读的操作。利用旋钮传感器和马达风扇模块，对eprom进行探究。

思路分析

Arduino要想实现断电数据也不会丢失是非常简单的，因其已经准备好了EEPROM，在编写程序时，首先引入该类库，再通过put,get方法即可实现对数据的写与读的操作，通过判断写入地址数值，可实现对要执行功能的保存。注意EEPROM的写入/擦除寿命为100000次左右，使用前一定要计算好自己的写入频率以确定Arduino的有效工作期限。



硬件连线



序号	名称	连线	备注
1	旋钮传感器	A0	
2	马达风扇模块	D6	



程序编程



- 1、EEPROM数据保存以字节形式，大小为1K，即512字节。在初始化中向addr中写入数据1。
- 2、读取EEPROM地址addr的数据，并且保存到item变量。
- 3、串口打印item的值，打开串口监视器查看。
- 4、对取到的值进行判断，如果是item等于写进地址addr的值，执行调节马达风扇速度大小代码。



1、头文件引入

```
#include <EEPROM.h>
```

2、定义变量

```
volatile int item;  
volatile int addr;  
volatile int moni;
```

3、setup函数

```
void setup(){  
    item = 0;  
    addr = 0;  
    moni = 0;  
    EEPROM.put(addr, 1); //向eprom地址addr写入数据1  
    Serial.begin(9600);  
}
```

4、loop函数

```
void loop(){  
    EEPROM.get(addr, item); //读取eprom地址addr的数据保存到item变量中  
    Serial.println(item);  
    if (item == 1){  
        moni = analogRead(A0); //读取模拟管脚A0的模拟值  
        analogWrite(6,(map(moni, 1, 1023, 20, 255))); //建立模拟输入和模拟输出的映射关系  
    }  
}
```

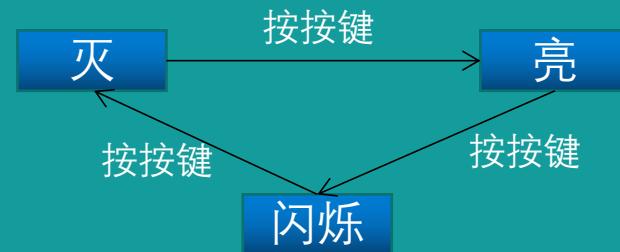


项目简析



项目任务

状态转换。灭、亮、闪烁分别是3个状态，状态间转换的条件叫状态转换条件，这种编程方法叫有限状态机（FSM），复杂的控制逻辑分解成有限个稳定状态，在每个状态上判断事件，用有限的状态，处理无穷的事务。利用按键传感器和闪灯模块，实现状态转换。

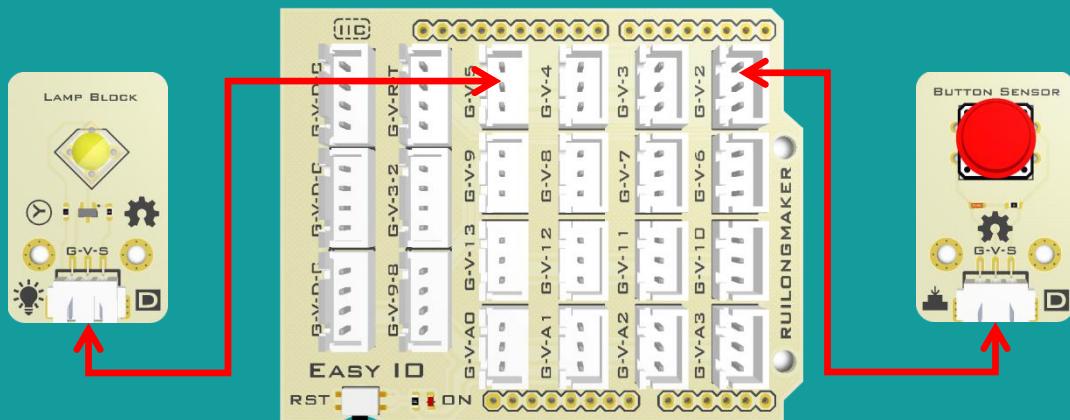


思路分析

Arduino以状态机的方式运行，需要用到一个重要的语句就是switch语句，是用来判断状态的，可以用一个标志数据来记录状态并且为其编号，判断这个编号，编写相应状态的执行语句。switch语句逻辑上可以看做if else语句。



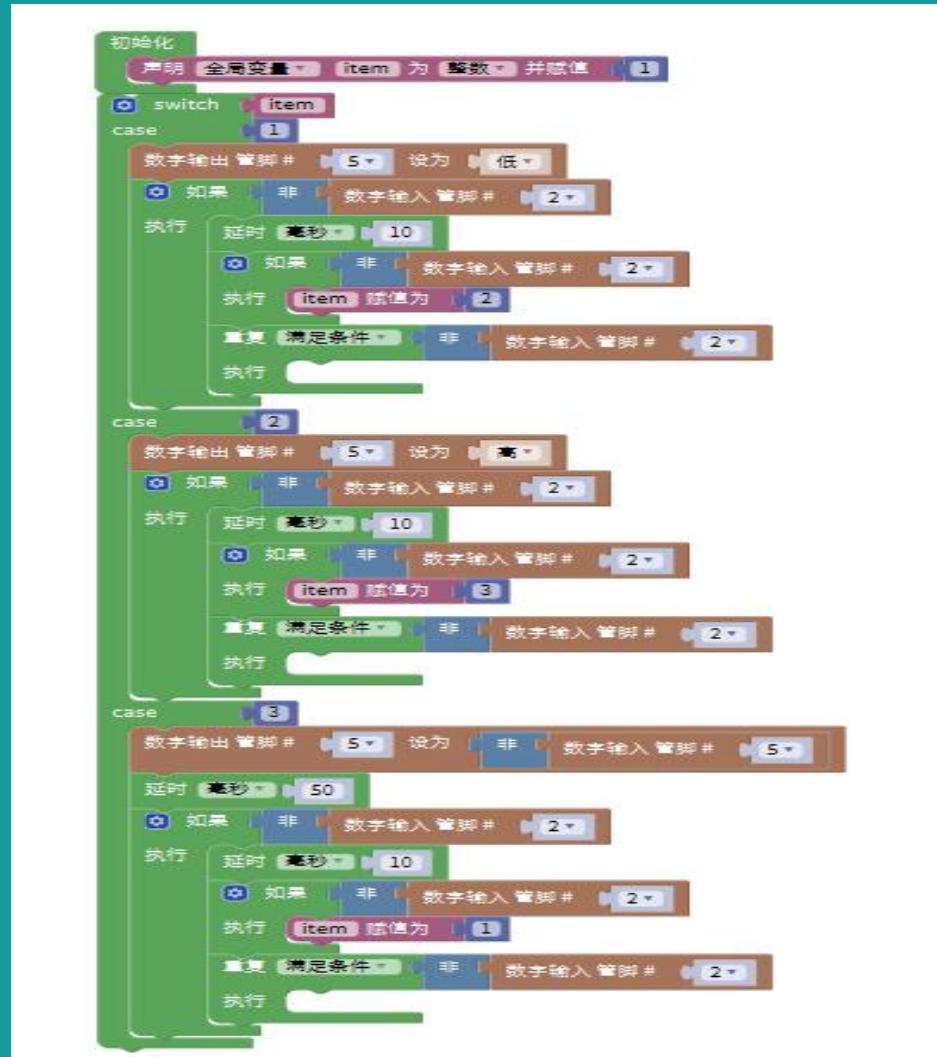
硬件连线



序号	名称	连线	备注
1	按钮传感器	D2	
2	闪灯模块	D5	



程序编程



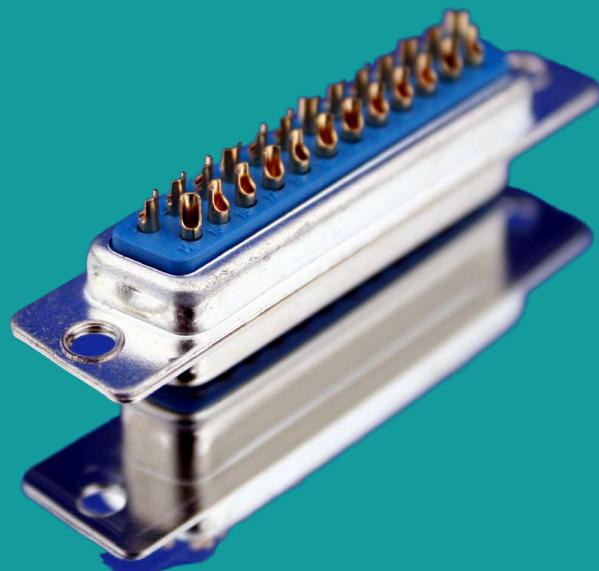
- 1、声明标志位变量。并赋值为1。
- 2、当item=1时，执行switch语句，闪灯模块熄灭，如果按下按钮传感器，item赋值为2。
- 3、当item=2时，执行switch语句，闪灯模块亮起，如果按下按钮传感器、item赋值为3。
- 4、当item=3时，执行switch语句，闪灯模块亮起，如果按下按钮传感器，item赋值为1。



```
volatile int item;//定义变量
void setup(){
    item = 1;
    pinMode(5, OUTPUT);//管脚5设置为输出模式
    pinMode(2, INPUT);//管脚2设置为输入模式
}
void loop(){
    switch (item) {
        case 1:
            digitalWrite(5,LOW);//向5号管脚写入低电平
            if (!digitalRead(2)) {
                delay(10);//按键消抖
                if (!digitalRead(2)) {
                    item = 2;
                }
                while (!digitalRead(2)) {
                }
            }
            break;
        case 2:
            digitalWrite(5,HIGH);
            if (!digitalRead(2)) {
                delay(10);
                if (!digitalRead(2)) {
                    item = 3;
                }
                while (!digitalRead(2)) {
                }
            }
            break;
        case 3:
            digitalWrite(5,!digitalRead(5));//向5号管脚写入读取到的5号管脚电平的相反值
            delay(50);
            if (!digitalRead(2)) {
                delay(10);
                if (!digitalRead(2)) {
                    item = 1;
                }
                while (!digitalRead(2)) {
                }
            }
            break;
    }
}
```



项目简析



项目任务

arduino实现串口的部件是USART,即通用同步/异步串行收发器，Arduino实现了硬串口和软串口两种形式的串口通信，并且都以类的形式进行管理。硬串口的操作类为HardwareSerial，定义于HardwareSerial.h源文件中，并对用户公开声明了Serial对象，用户在Arduino程序中直接调用Serial，就可实现串口通讯。

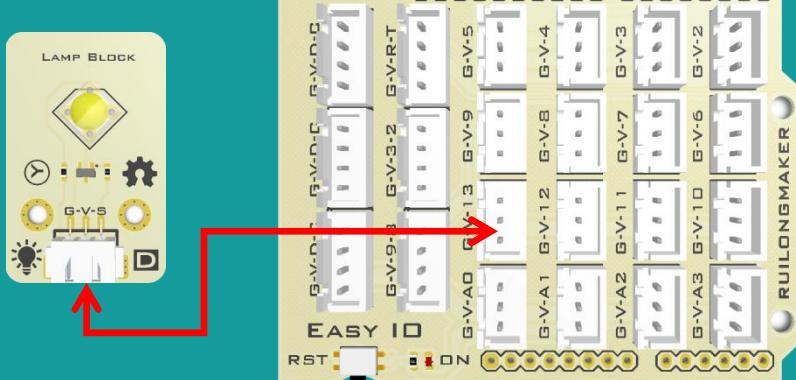
软串口的操作类为SoftwareSerial，定义于SoftwareSerial.h源文件中，但不像硬串口那样，源文件中并没有事先声明软串口对象，Arduino程序中需要手动创建软串口对象。利用硬件串口通信点亮或关闭闪灯模块。

思路分析

利用arduino硬件串口通信，通过串口监视器，发送1，实现闪灯模块亮起，发送0，闪灯模块熄灭。



硬件连线



序号	名称	连线	备注
1	闪灯模块	D13	



程序编程



- 1、设置串口波特率为9600。
- 2、如果串口有数据可以读取，调用对象Serial并调用read方法，获取数据。
- 3、判断数据，执行相应语句，实现对闪灯模块亮灭的切换。
- 4、打开串口监视器，发送数据。



1、定义变量

```
volatile char Light;
```

2、setup函数

```
void setup(){
    Serial.begin(9600); //设置串口比特率
    Light = '0';
    pinMode(13, OUTPUT); //管脚13设置为输出模式
}
```

3、loop函数

```
void loop(){
    if (Serial.available() > 0) { //如果串口有数据可读
        Light = Serial.read(); //light赋值为串口读取值
    }
    switch (Light) {
        case '1':
            digitalWrite(13,HIGH); //13管脚写入高电平
            break;
        case '0':
            digitalWrite(13,LOW);
            break;
    }
}
```



综合应用



项目简析



项目任务

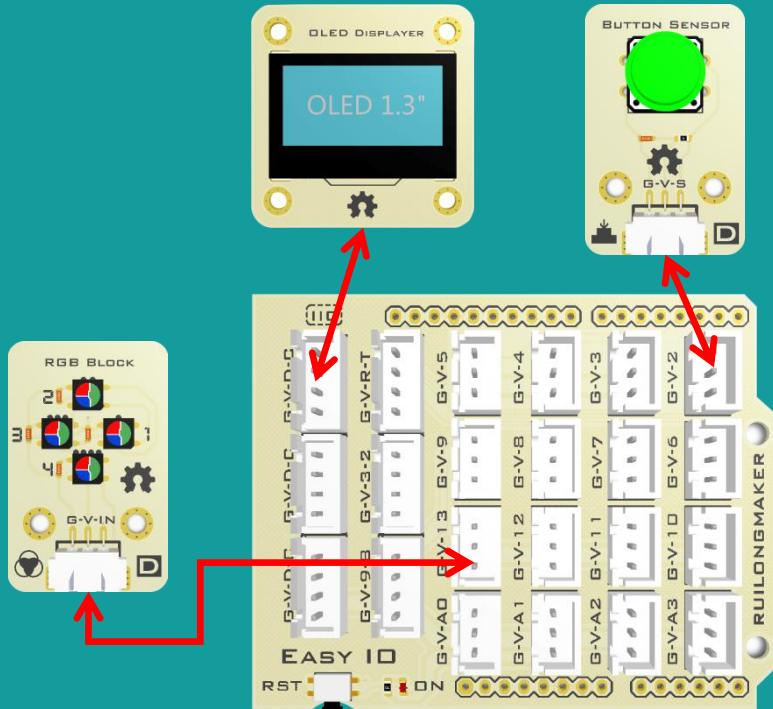
通过SSD显示器，RGB彩灯模块、按钮传感器制作一个“幸运大转盘”，按下按钮传感器，随机抽取奖励并且显示在SSD显示器上。

思路分析

利用循环对每个彩灯进行的R,G,B值进行清零、清零后循环点亮每一颗灯，创建随机数，随机显示某一颗灯，根据显示灯的亮起，SSD显示相应的奖励。



硬件连线



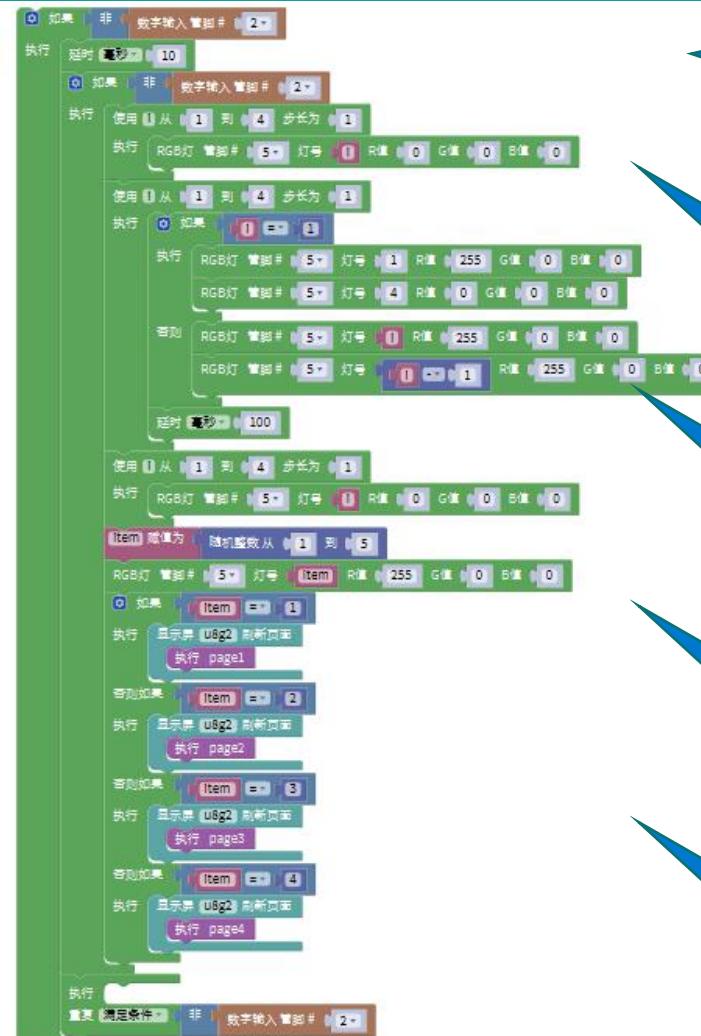
序号	名称	连线	备注
1	SSD显示器	GVDC	
2	按钮传感器	D2	
3	RGB彩灯模块	D13	



程序编程



SSD初始化及
显示内容



延时10ms用
于按键消抖

清除原有效果

4颗灯依次点
亮

随机显示某一
颗灯

依据随机数执
行相应的显示
页面



1、头文件引入

```
#include <U8g2lib.h>
#include <Wire.h>
#include <Adafruit_NeoPixel.h>
```

2、创建对象

```
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);
volatile int item;
Adafruit_NeoPixel rgb_ruilong_5(4,5, NEO_GRB + NEO_KHZ800);
```

3、显示页面

```
void page1() {
    u8g2.setFont(u8g2_font_timR08_tf); //设置字体
    u8g2.setFontPosTop();
    u8g2.setCursor(0,20); //在起点x=0,y=20显示文本。
    u8g2.print("Reward a pen"); //打印Reward a pen
}
```

```
void page2() {
    u8g2.setFont(u8g2_font_timR08_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(0,20);
    u8g2.print("Reward a Book");
}
```

```
void page3() {
    u8g2.setFont(u8g2_font_timR08_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(0,20);
    u8g2.print("Reward stationery");
}
```

```
void page4() {
    u8g2.setFont(u8g2_font_timR08_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(0,20);
    u8g2.print(" There is no reward");
}
```

4、setup函数

```
void setup(){
    u8g2.setI2CAddress(0x3C*2);
    u8g2.begin();
    item = 0;
    pinMode(2, INPUT);
    rgb_ruilong_5.begin();
    u8g2.enableUTF8Print();
}
```

5、loop函数

```
void loop(){
    if (!digitalRead(2)) {
        delay(10);
    }
    if (!digitalRead(2)) {
        for (int i = 1; i <= 4; i = i + (1)) {
            rgb_ruilong_5.setPixelColor(i-1, 0,0,0); //设置彩灯颜色
            rgb_ruilong_5.show(); //显示彩灯颜色
        }
        for (int i = 1; i <= 4; i = i + (1)) {
            if (i == 1) {
                rgb_ruilong_5.setPixelColor(1-1, 255,0,0);
                rgb_ruilong_5.show();
            }
            if (i == 2) {
                rgb_ruilong_5.setPixelColor(2-1, 255,0,0);
                rgb_ruilong_5.show();
            }
            if (i == 3) {
                rgb_ruilong_5.setPixelColor(3-1, 255,0,0);
                rgb_ruilong_5.show();
            }
            if (i == 4) {
                rgb_ruilong_5.setPixelColor(4-1, 255,0,0);
                rgb_ruilong_5.show();
            }
        }
        delay(100);
    }
    for (int i = 1; i <= 4; i = i + (1)) {
        rgb_ruilong_5.setPixelColor(i-1, 0,0,0);
        rgb_ruilong_5.show();
    }
    item = random(1, 5);
    rgb_ruilong_5.setPixelColor(item-1, 255,0,0);
    rgb_ruilong_5.show();
}
```



```
if (item == 1) {
    u8g2.firstPage();//刷新页面
    do
    { page1();
    }while(u8g2.nextPage());
} else if (item == 2)
{
    u8g2.firstPage();
    do{ page2();
    }while(u8g2.nextPage());
}
else if (item == 3) {
    u8g2.firstPage();
    do {
        page3();
    }while(u8g2.nextPage());
} else if (item == 4) {
    u8g2.firstPage();
    do{
        page4();

    }while(u8g2.nextPage());}}
do{
    }while(!digitalRead(2));}
```



项目简析



项目任务

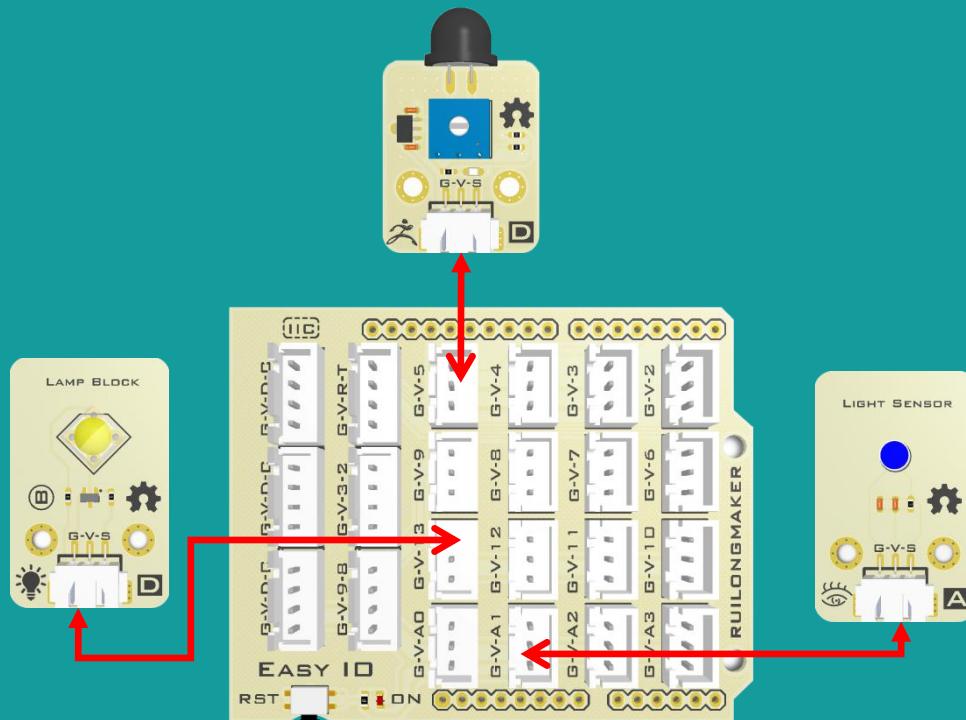
利用热释电红外传感器、光线传感器、彩灯模块制作一个楼道感应灯。

思路分析

热释电红外传感器是数字模拟器，光线传感器是模拟传感器，设定当光线传感器的模拟值小于200当量时，认定为黑夜状态（可以自行设定光线强度），当满足热释电红外传感器感受到人体时且光线传感器模拟值小于200，闪灯模块亮起，否则闪灯模块熄灭。



硬件连线



序号	名称	连线	备注
1	热释电红外传感器	D5	
2	闪灯模块	D13	
3	光线传感器	A1	



程序编程



- 1、串口打印A1管脚的模拟值
- 2、热释电红外传感器感受到人体且光线传感器模拟值小于200，闪灯模块亮起，延时1000ms(自行设定)，否则熄灭。



```
void setup(){
    Serial.begin(9600);
    /*通过pinMode函数，可以将Arduino的引脚配置为以下三种模式
    1、输出(OUTPUT)模式
        当引脚设置为输出 (OUTPUT) 模式时，引脚为低阻抗状态。
        这意味着Arduino可以向其它电路元器件提供电流
    2、输入(INPUT)模式
        当引脚设置为输入 (INPUT) 模式时，引脚为高阻抗状态
        此时该引脚可用于读取传感器信号或开关信号。
    3、输入上拉 (INPUT_PULLUP) 模式
        Arduino 微控制器自带内部上拉电阻。如果你需要使用该内部上拉电阻，
        可以通过pinMode()将引脚设置为输入上拉 (INPUT_PULLUP) 模式。
*/
pinMode(5, INPUT);
pinMode(13, OUTPUT);
}

void loop(){
    //打印模拟输入管脚A1的数值
    Serial.println(analogRead(A1));
    //&&与运算，两个表达式为真，才为真。
    if (digitalRead(5) && analogRead(A1) < 200) {
        //点亮闪灯模块
        digitalWrite(13,HIGH);
        delay(1000);
        //&&运算结果为假，熄灭闪灯模块
    } else {
        digitalWrite(13,LOW);
    }
}
```



项目简析



项目任务

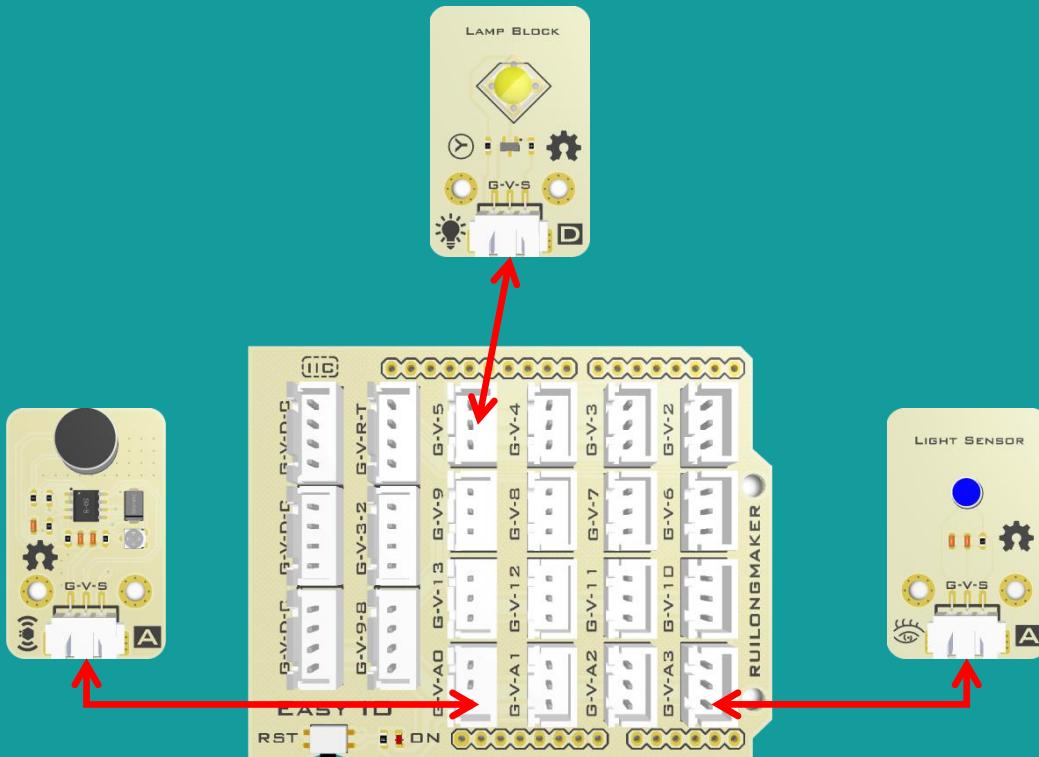
通过光线传感器、声音传感器、闪灯模块制作一个“声光控路灯”，当处于黑夜且感受到声音强度大于一定当量，闪灯模块亮起。

思路分析

光线传感器和声音传感器都是模拟传感器，其模拟数值的范围是0-1023，设定当光线传感器模拟数值小于300时为黑夜状态，处于黑夜状态且声音传感器模拟数值大于100时，闪灯模块亮起，延时5秒，闪灯模块熄灭。模拟值设定的阈值和闪灯延时时间可自行设定。



硬件连线



序号	名称	连线	备注
1	闪灯模块	D5	
2	声音传感器	A0	
3	光线传感器	A3	



程序编程



```
void setup(){
    Serial.begin(9600);
    pinMode(5, OUTPUT); //管脚5设置为输出模式
}
void loop(){
    Serial.println(analogRead(A0)); //串口打印读取到的管脚A0模拟值
    Serial.println(' ');
    Serial.println(analogRead(A3));
    if (analogRead(A3) < 300) {
        if (analogRead(A0) > 100) {
            digitalWrite(5,HIGH); //向5号管脚写入高电平
            delay(5000);
            digitalWrite(5,LOW); //向5号管脚写入低电平
        }
    }
}
```

- 1、串口打印声音传感器，光线传感器模拟数值。
- 2、光线传感器模拟值小于300，声音传感器模拟值大于100，闪灯模块亮起，延时5秒钟，闪灯模块熄灭。
- 3、不满足上述条件其一，闪灯模块熄灭。



项目简析



项目任务

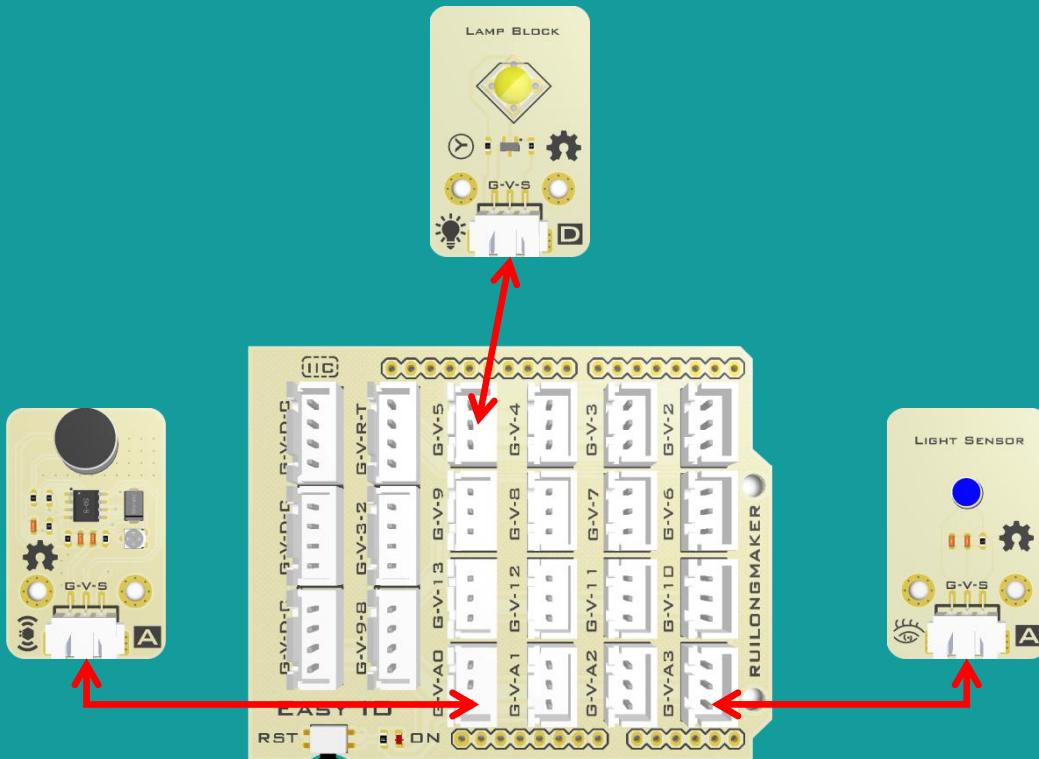
通过光线传感器、声音传感器、闪灯模块模拟车位检测，当汽车接近车位时、声音传感器检测到较强的声音强度，汽车到达停车位，地面安置的光线传感器感知到上方光线被遮挡，汽车停入车位，声音传感器感知到减弱的声音强度，闪灯模块亮起，表示该车位已经被占据。

思路分析

光线传感器和声音传感器都是模拟传感器，其模拟数值的范围是0-1023，设定阈值，声音传感器感知到的模拟值大于300，汽车正在接近停车位，安装在地面的光线传感器感受到光线强度小于100，且声音传感器模拟值大于300，该车正在进入车位，进入车位后，汽车熄火，声音强度小于100，且光线强度小于100，代表该车为被占据，闪灯模块亮起。



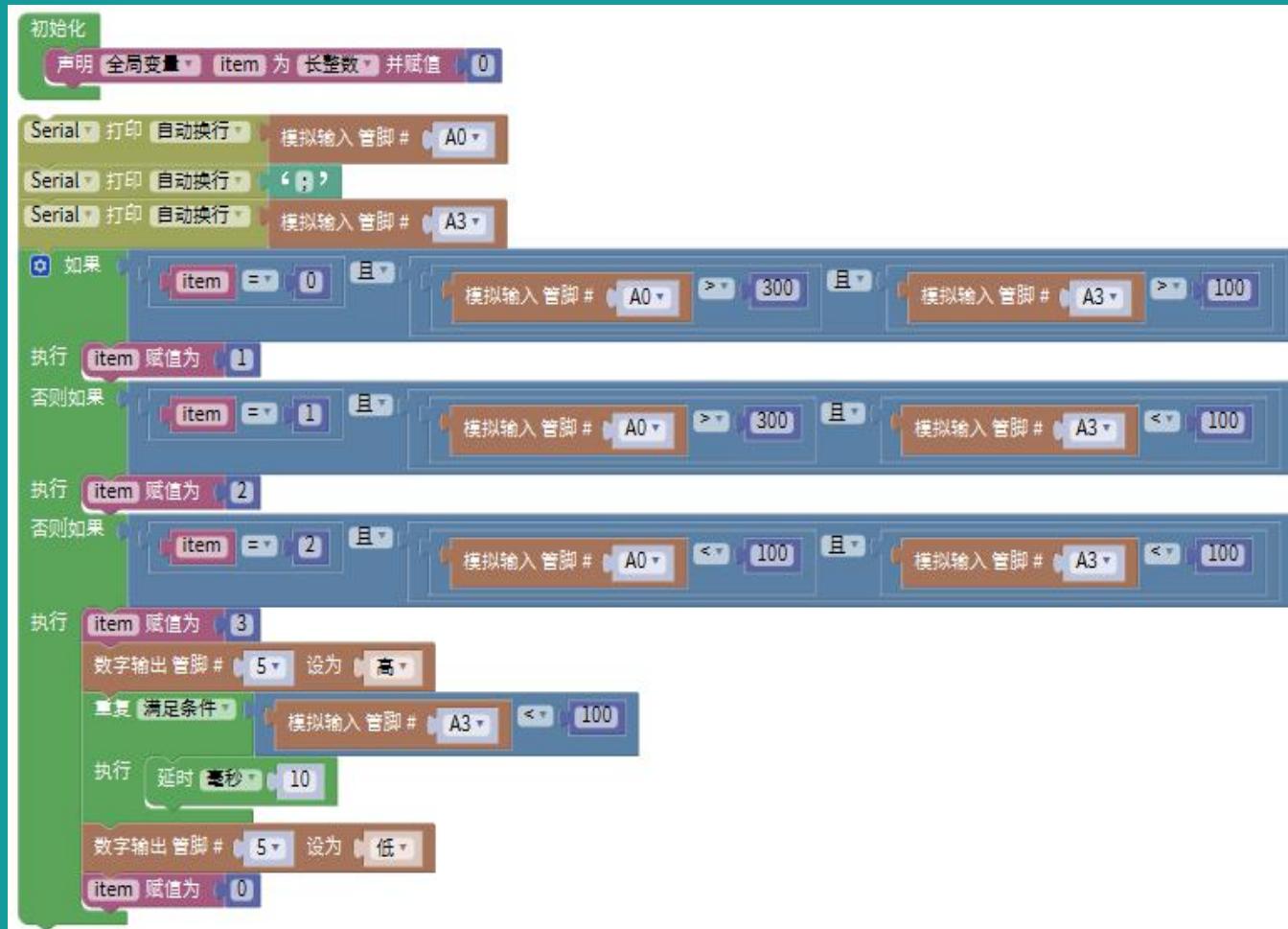
硬件连线



序号	名称	连线	备注
1	闪灯模块	D5	
2	声音传感器	A0	
3	光线传感器	A3	



程序编程



- 1、初始化变量item变量并赋值为0，代表车位为空。
- 2、通过串口打印声音传感器和光线传感器模拟值。
- 3、当item=0时，如果检测到的声音强度大于300，光线强度大于100，item赋值为1，表示车辆正在进入车库。
- 4、当item=1时，如果检测到声音强度大于300，光线强度小于100，item赋值为2，表示车辆正在进入车位。
- 5、当item=2时，如果检测到的声音强度小于100，光线强度小于100，item赋值为3，闪灯模块亮起、表示车辆熄火，停入该车位。
- 6、检测到的光线强度持续小于100，则该车位一直被占据，直到不满足这一条件，LED灯熄灭，item赋值为0，车位为空。



1、定义变量

```
volatile long item;
```

2、setup函数

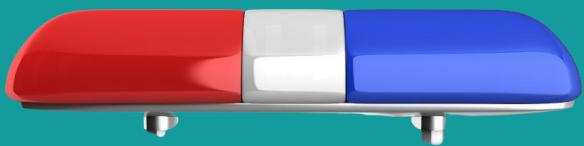
```
void setup(){
    item = 0;//item初始化为0，代表车位为空
    Serial.begin(9600);//设置波特率
    pinMode(5, OUTPUT);//管脚5设置为输出模式，用于点亮闪灯模块
}
```

3、loop函数

```
void loop(){
    Serial.println(analogRead(A0));//串口打印声音传感器数值
    Serial.println(';');
    Serial.println(analogRead(A3));
    if (item == 0 && (analogRead(A0) > 300 && analogRead(A3) > 100)) {
        item = 1;
    } else if (item == 1 && (analogRead(A0) > 300 && analogRead(A3) < 100)) {
        item = 2;
    } else if (item == 2 && (analogRead(A0) < 100 && analogRead(A3) < 100)) {
        item = 3;
        digitalWrite(5,HIGH);//向5号管脚写入高电平，闪灯模块亮起
        while (analogRead(A3) < 100) {
            delay(10);
        }
        digitalWrite(5,LOW);//向5号管脚写入低电平，闪灯模块熄灭
        item = 0;
    }
}
```



项目简析



项目任务

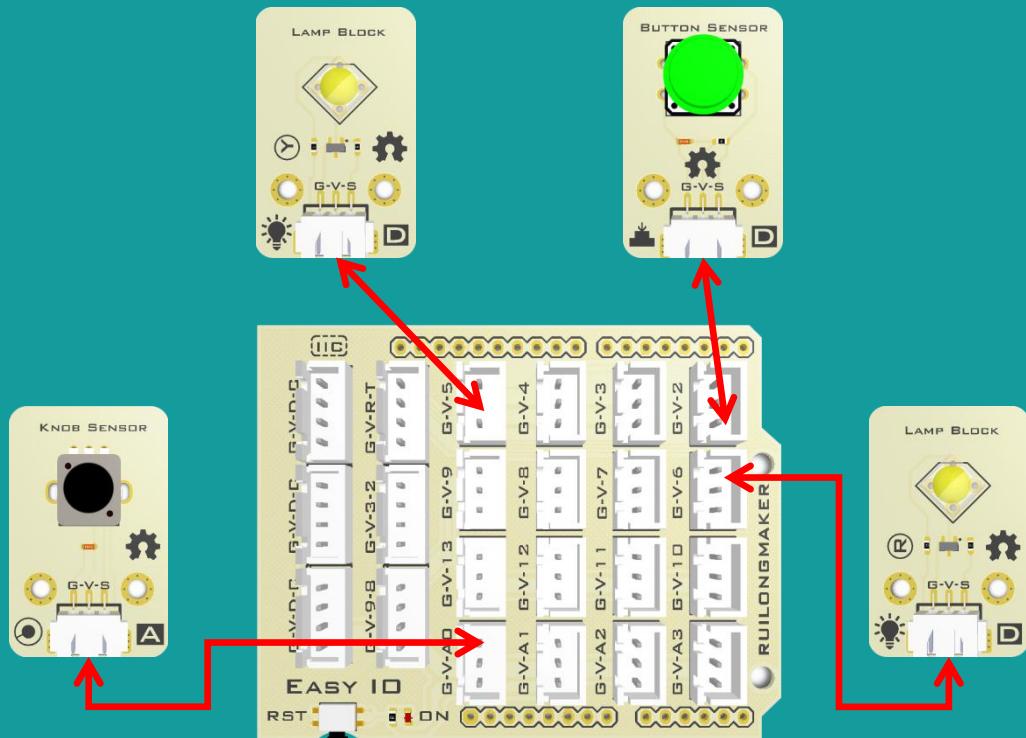
特种车辆的警报灯往往有着不同的颜色和闪烁效果，救护车用的是全蓝颜色，消防车用的是全红颜色，警车用的是红蓝颜色，而这些警灯都有着至少两种的闪烁效果，利用旋钮传感器、按钮传感器、闪灯模块（红色、黄色）模拟警报灯两种不同的闪烁效果。

思路分析

按钮传感器连接管脚2并设置软件中断模式，当按下按钮时，实现对警报灯的开关切换，设定一个阈值，当旋钮传感器的模拟值大于这个阈值，设定一个时间间隔，否则，设定另外一个时间间隔，以此进行闪灯模块闪烁效果的切换。



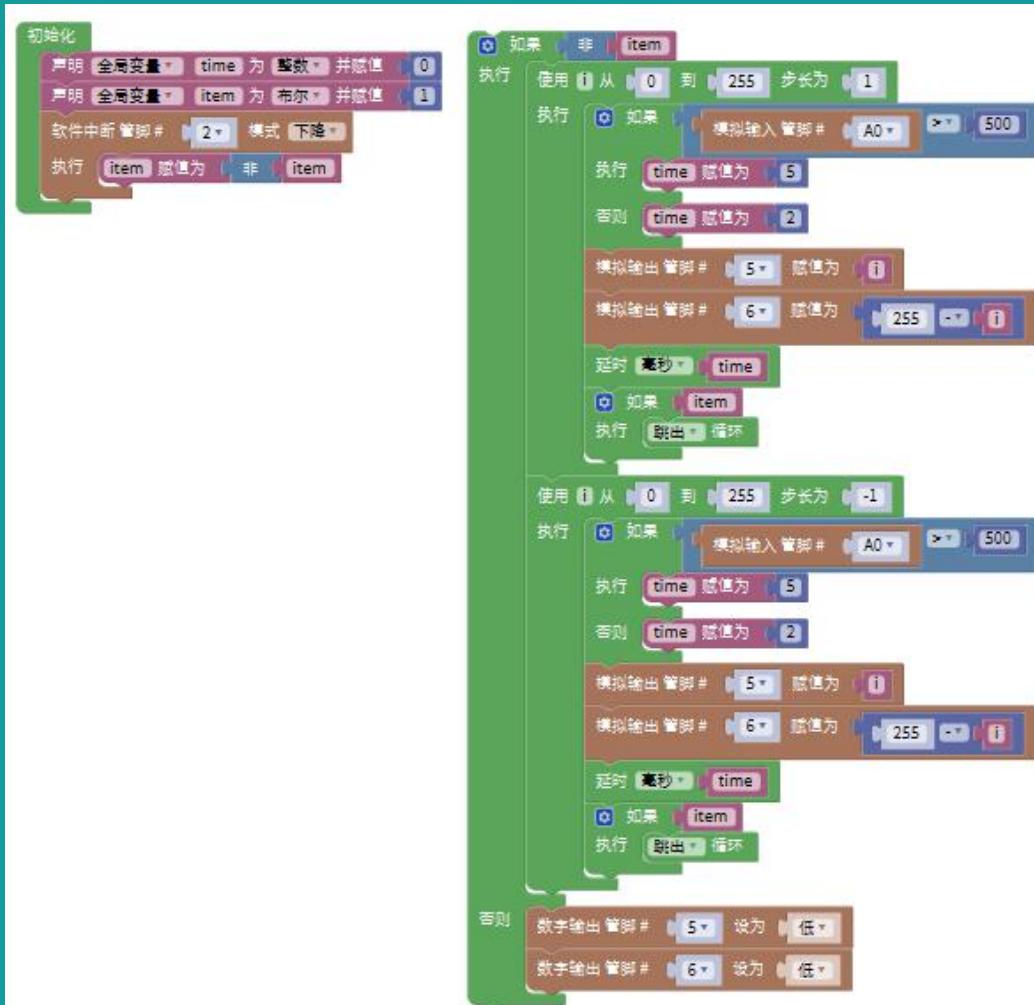
硬件连线



序号	名称	连线	备注
1	按钮传感器	D2	
2	闪灯模块 (黄色)	D5	
3	闪灯模块 (红色)	D6	
4	旋钮传感器	A0	



程序编程



- 1、初始化变量item、time赋值为0。
- 2、管脚2连接按钮传感器设置为软件中断模式，按下按钮，报警灯开关切换。
- 3、旋钮传感器连接管脚A0，读取到的模拟值大于500，time赋值为5，否则time赋值为2。
- 4、使用循环对闪灯模块亮度进行调节。



1、引入库文件

```
#include <PinChangeInt.h>
```

2、定义变量

```
volatile int time;  
volatile boolean item;
```

3、软件中断函数

```
void attachPinInterrupt_fun_2(){  
    item = !item;  
}
```

4、setup函数

```
void setup(){  
    time = 0;  
    item = 1;  
    pinMode(2, INPUT); //管脚2设置为输入模式  
    //类外attachInterrupt函数，参数管脚号、软件中断函数、中断模式为电平下降  
    PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,FALLING);  
    pinMode(5, OUTPUT); //管脚5设置为输出模式  
    pinMode(6, OUTPUT);  
}
```

5、loop函数

```
void loop(){  
    if (!item) {  
        for (int i = 0; i <= 255; i = i + (1)) {  
            if (analogRead(A0) > 500) //读取管脚A0模拟值  
                time = 5;  
            } else {  
                time = 2;  
            }  
            analogWrite(5,i);  
            analogWrite(6,(255 - i)); //向管脚6写入模拟值 (255-i)  
            delay(time);  
            if (item) {  
                break;  
            }  
        }  
        for (int i = 0; i <= 255; i = i + (-1)) {  
            if (analogRead(A0) > 500) {  
                time = 5;  
            } else {  
                time = 2;  
            }  
            analogWrite(5,i);  
            analogWrite(6,(255 - i));  
            delay(time);  
            if (item) {  
                break;  
            }  
        }  
    } else {  
        digitalWrite(5,LOW); //向管脚5写入低电平  
        digitalWrite(6,LOW);  
    }  
}
```



项目简析



项目任务

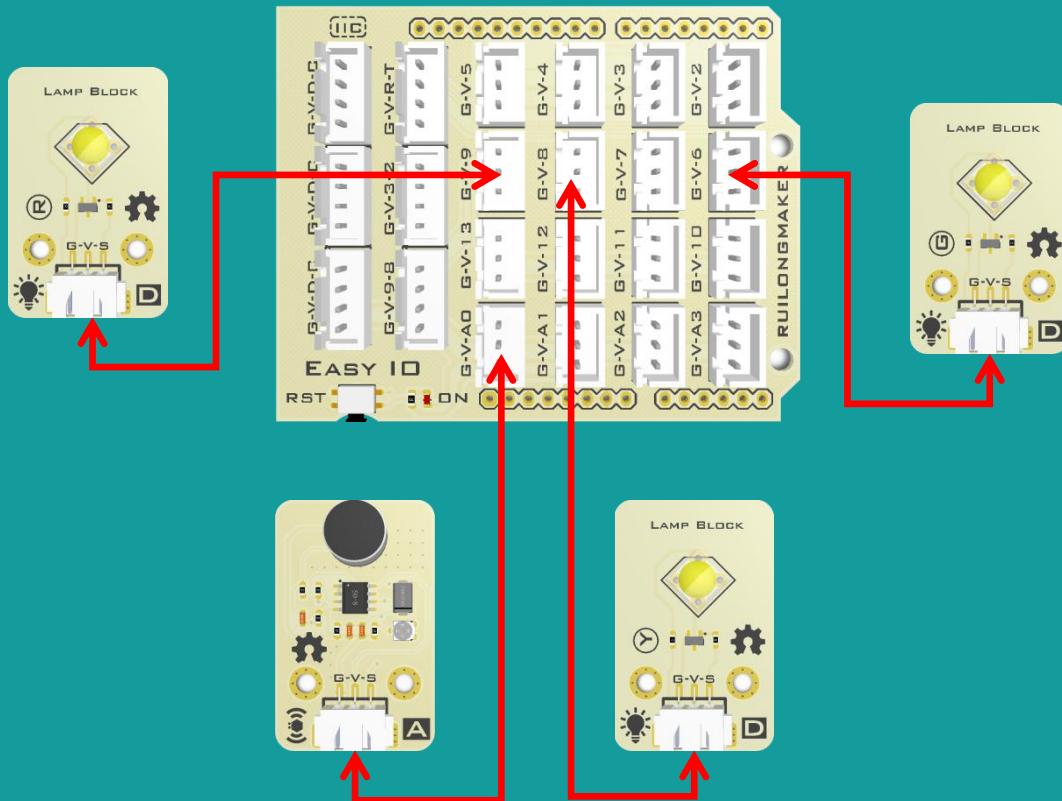
通过闪灯模块及声音传感器制作一个噪声监测仪，该检测仪通过声音传感器获取外界声音，根据声音的大小点亮不同颜色的闪灯模块。

思路分析

初始化变量用于保存声音传感器获取到的模拟值，当该模拟值小于300(自行设定) 亮绿灯，当该模拟值小于600,亮黄灯，否则亮红灯。



硬件连线



序号	名称	连线	备注
1	声音传感器	A0	
2	闪灯模块 (绿色)	D6	
3	闪灯模块 (黄色)	D8	
4	闪灯模块 (红色)	D9	



程序编程



- 1、初始化变量item并赋值为0。
- 2、声音传感器获取到的模拟值赋值给item。
- 3、如果item小于300，绿色闪灯模块亮起，代表噪声小。
- 4、如果item小于600，黄色闪灯模块亮起，代表噪声适度。
- 5、否则红色闪灯模块亮起代表噪声大。



1、定义变量

```
volatile int item;
```

2、setup函数

```
void setup(){
    item = 0;
    Serial.begin(9600);
    pinMode(6, OUTPUT); //管脚6设置为输出模式
    pinMode(8, OUTPUT);
    pinMode(9, OUTPUT);
}
```

3、loop函数

```
void loop(){
    item = analogRead(A0);
    Serial.println(item); //串口打印item值
    if (item < 300) {
        digitalWrite(6,HIGH); //向管脚6写入高电平
        digitalWrite(8,LOW);
        digitalWrite(9,LOW);

    } else if (item < 600) {
        digitalWrite(6,LOW);
        digitalWrite(8,HIGH);
        digitalWrite(9,LOW);
    } else {
        digitalWrite(6,LOW);
        digitalWrite(8,LOW);
        digitalWrite(9,HIGH);

    }
}
```



项目简析



项目任务

LM35温度传感器感知周围环境温度，然后以电压的方式输出，需要通过运算方式，将模拟电压值转化为直接读取的摄氏度温度值，模拟输入的数值0-1023对应电压值为0-5000mv,单位电压值为5000除以1024 (mv),TM35传感器每10mv对应1摄氏度，实际温度值则可以表示为：

温度值=模拟值 *(5/10.24)，睿龙创客工厂库直接转化为了模拟值*0.488，可直接使用。

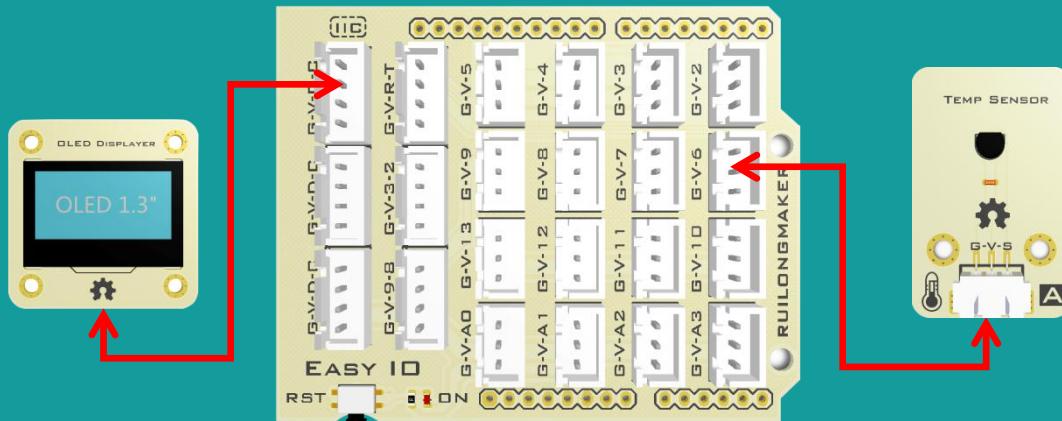
通过TM35温度传感器和SSD1306显示屏，制作“穿衣提醒”，通过温度传感器对环境温度的获取，提示你应该穿什么衣服。

思路分析

通过温度转换关系将读取到的电压模拟值转化为可读取的摄氏度温度值，划分温度范围，判断可读取的摄氏度温度值所处范围，执行相应语句，并在SSD1306显示屏上显示该穿什么衣服。



硬件连线



序号	名称	连线	备注
1	SSD1306	GVDC	
2	TM35温度传感器	A0	



程序编程

The image shows a Scratch-like programming environment for the Arduino Easy Learning Kit. The workspace is divided into two main sections: 'Initialization' on the left and 'Program' on the right.

Initialization: This section contains the setup code for the SSD1306 display and the temperature sensor (LM35). It includes the declaration of a global variable 'temp' as a decimal number with a value of 0. A note indicates that the睿龙创客工厂库 has already converted the temperature from Celsius to Fahrenheit, so it can be used directly.

Program: This section contains three pages of穿衣提醒 logic:

- page1:** Handles temperatures between 0 and 15 degrees. It displays the current temperature and a message: "wear a down jacket".
- page2:** Handles temperatures between 15 and 25 degrees. It displays the current temperature and a message: "wear a jacket".
- page3:** Handles temperatures above 25 degrees. It displays the current temperature and a message: "wear a T-shirt".

A large blue callout bubble points from the text in the Initialization section to the 'temp' variable in the 'page1' script, indicating that the variable is ready for use.

睿龙创客工厂库已
经进行了温度转换，
可直接使用



1、引入库文件

```
#include <U8g2lib.h>
#include <Wire.h>
```

2、创建对象

```
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);
```

3、定义变量

```
volatile float temp;
```

4、显示页面

```
void page1() {
/*设置字体*/
    u8g2.setFont(u8g2_font_timR10_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(0,20);//在坐标 (0, 20) 打印
    u8g2.print(temp);//打印温度
    u8g2.setCursor(0,40);//在坐标 (0, 40) 打印温度
    u8g2.print("wear a down jacket");//打印wear a down jacket
}
void page2() {
    u8g2.setFont(u8g2_font_timR10_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(0,20);
    u8g2.print(temp);
    u8g2.setCursor(0,40);
    u8g2.print("wear a jacket");
}
void page3() {
    u8g2.setFont(u8g2_font_timR10_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(0,20);
    u8g2.print(temp);
    u8g2.setCursor(0,40);
    u8g2.print("wear a T-shirt");
}
```

4、setup函数

```
void setup(){
    u8g2.setI2CAddress(0x3C*2);
    u8g2.begin();
    temp = 0;
    u8g2.enableUTF8Print();

    Serial.begin(9600);
}
```

5、loop函数

```
void loop(){
    temp = analogRead(A0)*0.488 //temp=模拟值*5/10.24=模拟值*0.488
    Serial.println(temp);
    if (temp < 15) {
        u8g2.firstPage();//刷新页面
        do
        {
            page1();//执行page1
        }while(u8g2.nextPage());

    } else if (temp >= 15 && temp < 25) {
        u8g2.firstPage();
        do
        {
            page2();
        }while(u8g2.nextPage());
    } else if (temp >= 25) {
        u8g2.firstPage();
        do
        {
            page3();
        }while(u8g2.nextPage());
    }
}
```



项目简析



项目任务

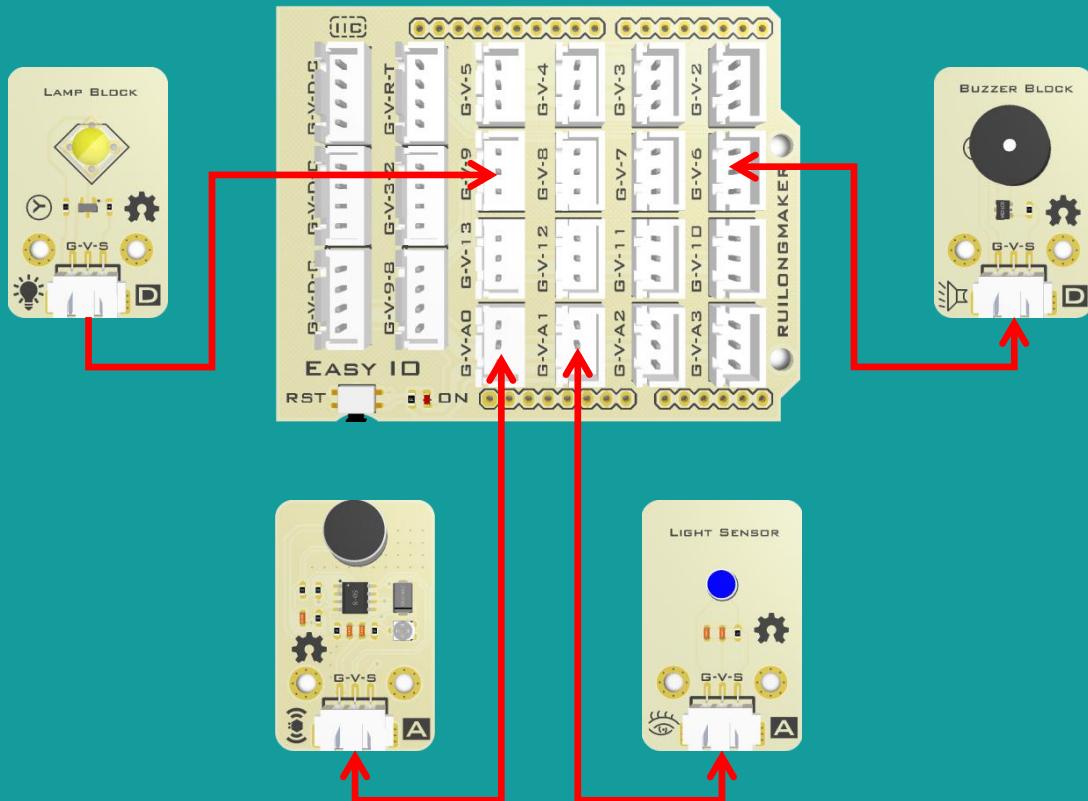
通过声音传感器、光线传感器、蜂鸣器模块、闪灯模块制作一根电子蜡烛，实现用闪灯模块模拟蜡烛；强光照射模拟“点亮蜡烛”的过程；用嘴吹气模拟“吹灭蜡烛”的过程；闪灯模块亮起，光线一闪一闪模拟真实蜡烛；闪亮的过程伴随歌曲。

思路分析

串口打印当前声音强度和光线强度，确定合适范围，定义播放音乐函数，用于蜂鸣器播放音乐，闪灯模块随音乐跳动，在函数中使用轮询，检查声音强度是否大于设定值，大于返回初始化，停止播放音乐，闪灯模块熄灭，当光线强大于设定值时，开始播放音乐，播放完成后自动停止。



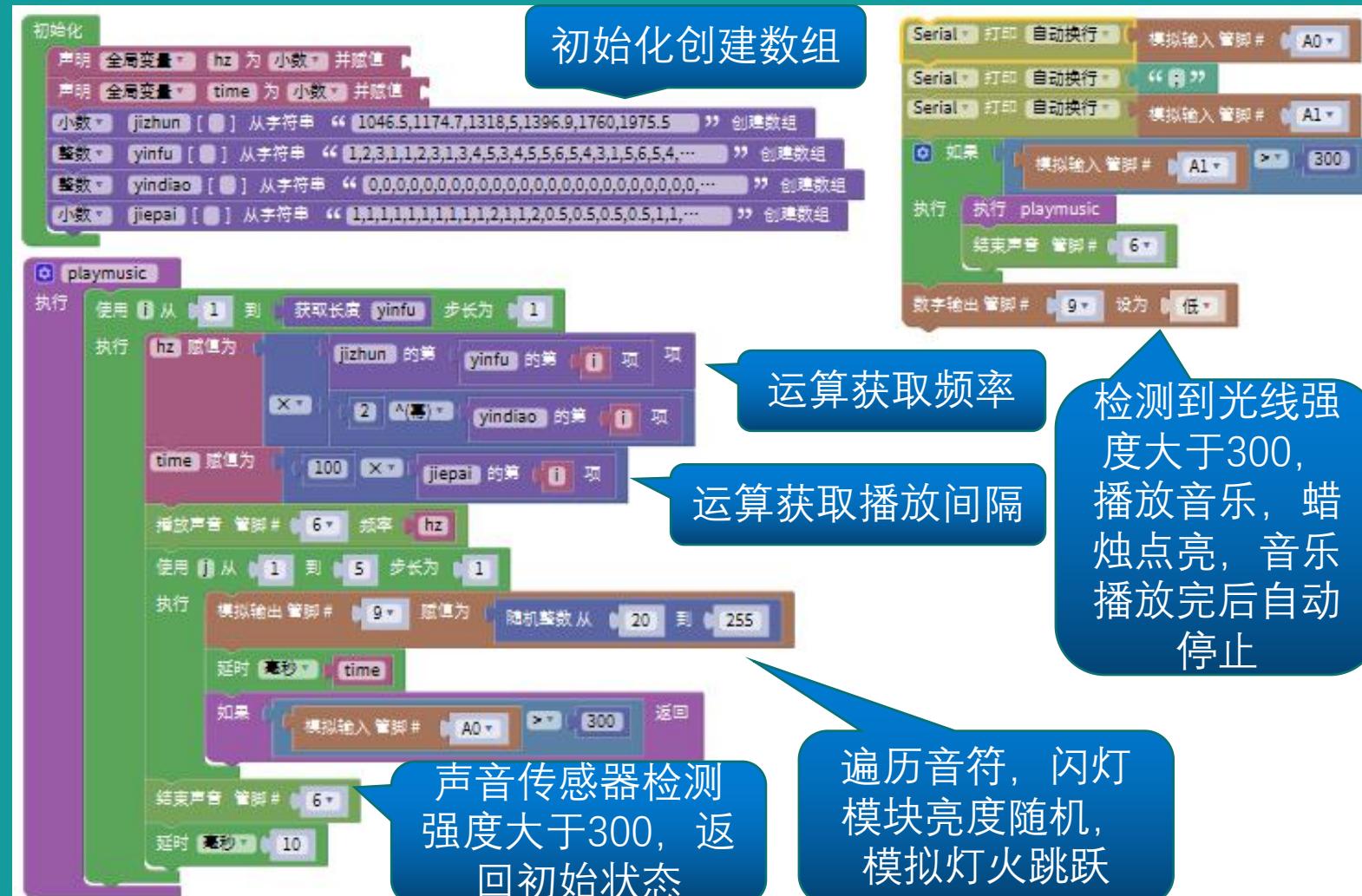
硬件连线



序号	名称	连线	备注
1	蜂鸣器模块	D6	
2	闪灯模块	D9	
3	声音传感器	A0	
4	光线传感器	A1	



程序编程





1、声明变量

```
volatile float hz;  
volatile float time;
```

2、创建数组

```
//基准音  
float jizhun[]={1046.5,1174.7,1318.5,1396.9,1760,1975.5};  
//音符  
int yinfu[]={1,2,3,1,1,2,3,1,3,4,5,3,4,5,5,6,5,4,3,1,5,6,5,4,3,1,2,5,1,2,5,1};  
//音调  
int yindiao[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,-1,0};  
//节拍  
float jiepai[]={1,1,1,1,1,1,1,1,2,1,1,2,0.5,0.5,0.5,0.5,1,1,0.5,0.5,0.5,0.5,1,1,1,2,1,2};
```

3、播放音乐函数

```
void playmusic() {  
//sizeof(yinfu)/sizeof(yinfu[0]) 获取数组长度  
for (int i = (1); i <= (sizeof(yinfu)/sizeof(yinfu[0])); i = i + (1)) {  
//运算获取播放声音频率  
hz = jizhun[(int)(yinfu[(int)(i - 1)] - 1)] * pow(2, yindiao[(int)(i - 1)]);  
//运算获取播放音乐间隙  
time = 100 * jiepai[(int)(i - 1)];  
//调用tone函数，参数为（管脚号， 播放频率  
tone(6,hz);  
for (int j = 1; j <= 5; j = j + (1)) {  
//9号管脚写入模拟值为20-255的随机数  
analogWrite(9,(random(20, 255));  
delay(time);  
//读取A0管脚的模拟值  
if (analogRead(A0) > 300) {  
return;  
}  
}  
noTone(6); delay(10); }
```

4、setup函数

```
void setup(){  
hz = 0;  
time = 0;  
Serial.begin(9600);  
//6号管脚设置为输出模式  
pinMode(6, OUTPUT);  
//9号管脚设置为输出模式  
pinMode(9, OUTPUT);  
}
```

5、loop函数

```
void loop(){  
//串口打印管脚A0模拟值  
Serial.println(analogRead(A0));  
Serial.println(";" );  
//串口打印管脚A1模拟值  
Serial.println(analogRead(A1));  
if (analogRead(A1) > 300) {  
//调用播放音乐函数  
playmusic();  
//调用noTone函数，功能为停止播放声音  
noTone(6);  
}  
digitalWrite(9,LOW);  
}
```



项目简析



项目任务

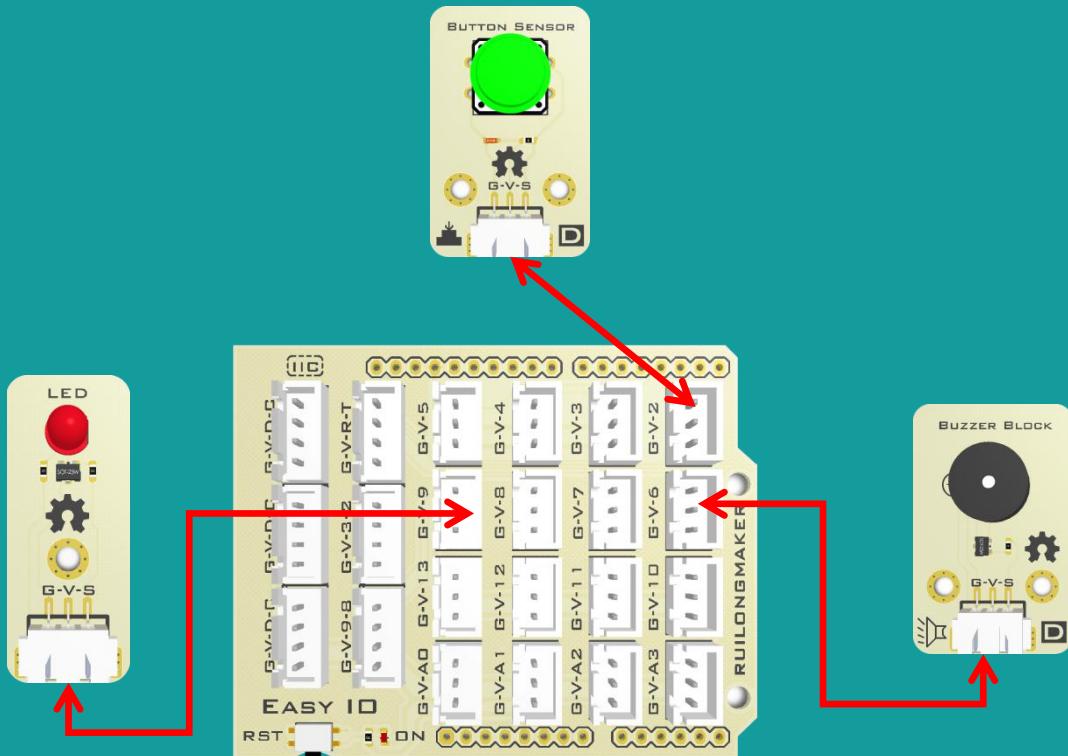
通过蜂鸣器模块、按钮传感器、LED灯模块制作一个流光溢彩音乐盒。

思路分析

初始化变量，并创建数组，设置管脚2为中断，模式为下降，定义播放音乐的函数，通过运算得到声音的频率及间隔，建立频率与LED灯亮度的映射关系，按下按钮，播放声音，LED灯开始闪烁，整首音乐播放完自动停止。播放声音过程中按下按钮，停止播放。



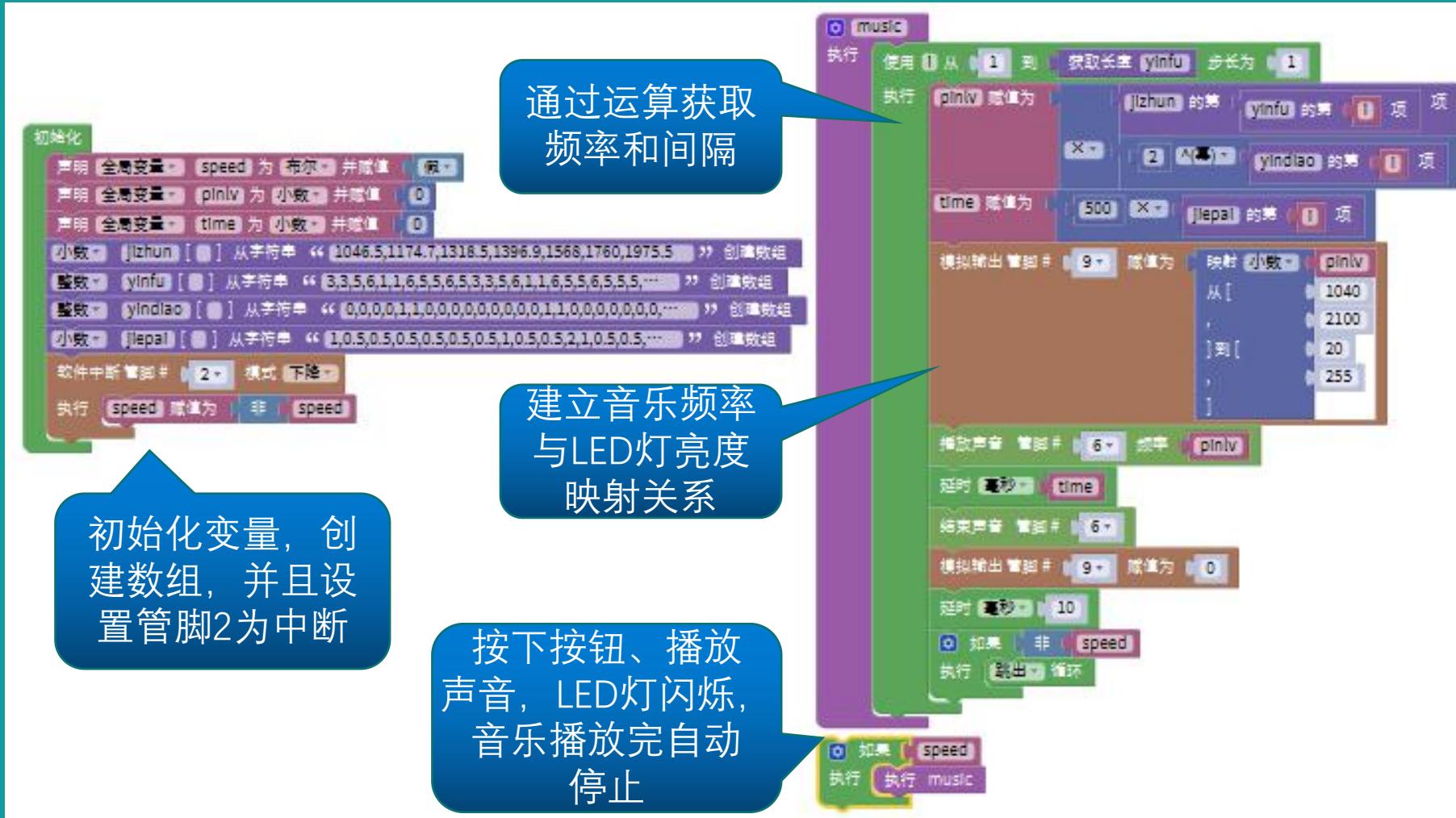
硬件连线



序号	名称	连线	备注
1	蜂鸣器模块	D6	
2	LED模块	D9	
3	按钮传感器	D2	



程序编程





1、头文件引入

```
#include <PinChangeInt.h>
```

2、定义变量

```
volatile boolean speed;
volatile float pinlv;
volatile float time;
```

3、数组创建

```
float jizhun[]={1046.5,1174.7,1318.5,1396.9,1568,1760,1975.5}//基准音
int yinfu[]={3,3,5,6,1,1,6,5,5,6,5,3,3,5,6,1,1,6,5,5,6,5,5,5,3,5,6,6,5,3,2,3,5,3,2,1,1,2,1};//音符
int yindiao[]={0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};//音调
float
jiepai[]={1,0.5,0.5,0.5,0.5,0.5,0.5,1,0.5,0.5,2,1,0.5,0.5,0.5,0.5,1,0.5,0.5,0.5,2,1,1,1,0.5,0.5,1,1,2,1,0.5,0.5,1,0.5,0.5,1};//节拍
```

4、映射函数

```
float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{ return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;}
```

5、播放音乐函数

```
void music(){
for (int i = 1; i <= (sizeof(yinfu)/sizeof(yinfu[0])); i = i + (1)) {
    pinlv = jizhun[(int)(yinfu[(int)(i - 1)] - 1)] * pow(2, yindiao[(int)(i - 1)]);
    time = 500 * jiepai[(int)(i - 1)];
    analogWrite(9,(mapfloat(pinlv, 1040, 2100, 20, 255)));
    tone(6,pinlv);
    delay(time);
    noTone(6);
    analogWrite(9,0);
    delay(10);
    if (!speed) {
        break; }}}
```

6、软件中断函数

```
void attachPinInterrupt_fun_2()
{
    speed = !speed;
}
```

7、setup函数

```
void setup()
{
    pinMode(6, OUTPUT);//管脚6设置为输出模式
    speed = false;
    pinlv = 0;
    time = 0;
    pinMode(2, INPUT);//管脚2设置为输入模式
    PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,FALLING);
//PCintPort类外函数attachInterrupt,参数2为管脚号, 参数attachPinInterrupt_fun_2,软件中断函数,
FALLING表示模式, 当发生中断时, 电平的改变模式。
}
```

8、loop函数

```
void loop()
{
    if (speed) {
        music();
    }
}
```



项目简析



项目任务

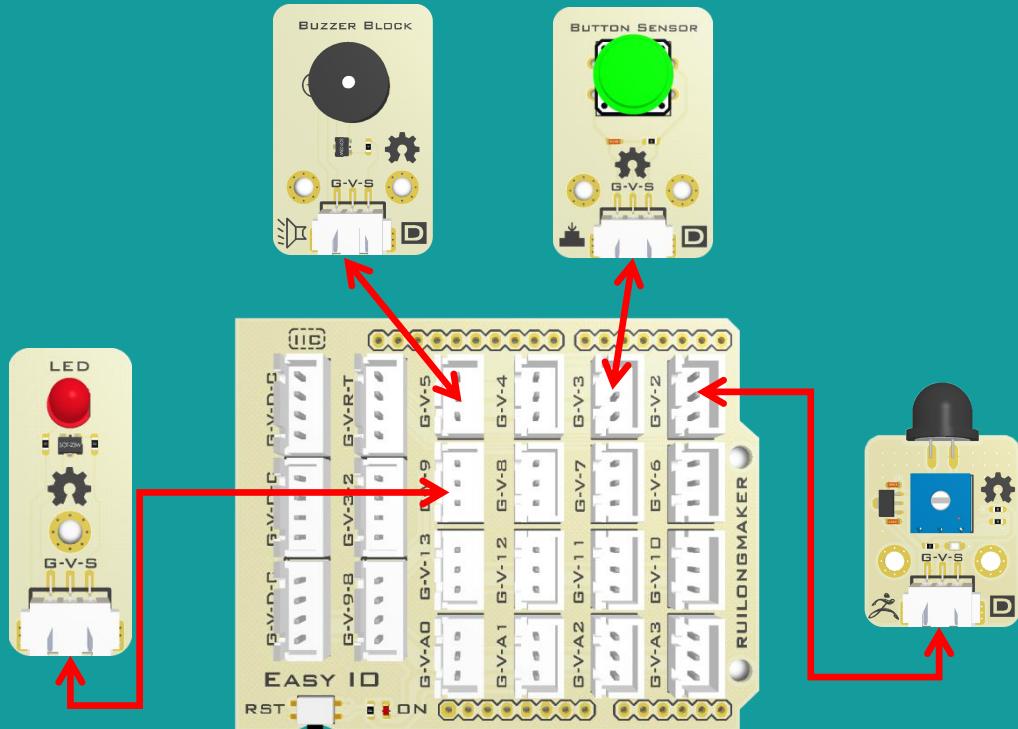
小偷往往会在夜晚通过窗户进入房间，进行违法犯罪活动。利用热释电传感器、蜂鸣器模块、按钮传感器、LED模块制作防扒窗报警器，报警器安装在窗户外侧，当产生热释电传感器检测到人体经过时，发出警报，并且人为的按下按钮，报警才会停止、LED灯熄灭。

思路分析

热释电传感器设置中断连接D2、按钮传感器设置中断连接D3，编写警报声函数，利用三角函数运算获取不同数值，来改变声音的频率，从而模拟警报声，当热释电传感器检测到持续人体经过时，蜂鸣器模块发出警报，点亮LED灯闪烁，按下按钮，触发中断，结束警报声，LED灯停止闪烁。



硬件连线



序号	名称	连线	备注
1	热释电传感器	D2	
2	按钮传感器	D3	
3	蜂鸣器模块	D5	
4	LED模块	D9	



程序编程



- 1、初始化变量并赋值为0。
- 2、管脚2连接热释电传感器，设置中断，热释电传感器默认为低电平（0），中断模式选择上升，管脚3连接按钮传感器，设置中断，按钮传感器默认高电平(1),中断模式选择下降。
- 3、定义sound函数，利用一个三角函数运算获取不同数值，来改变声音的频率，从而模拟警报声
- 4、如果热释电传感器检测到人体经过，发出警报声，LED灯开始闪烁。
- 5、按下按钮，触发中断，结束播放警报声，熄灭LED灯闪烁停止。



1、头文件引入

```
#include <PinChangeInt.h>
```

2、定义变量

```
volatile boolean item;  
volatile int hz;
```

3、中断函数

```
void attachPinInterrupt_fun_2()  
{  
    item = true;  
}
```

```
void attachPinInterrupt_fun_3()  
{  
    item = false;  
}
```

4、扬声器函数

```
void sound()  
{  
    for (int i = 1; i <= 180; i = i + (1)) {  
        hz = 2000 + round(sin(i / 180.0 * 3.14159) * 1000); //运算获取播放频率  
        tone(5,hz); //5号管脚连接蜂鸣器模块，播放频率为hz的音频  
        delay(10);  
    }  
}
```

5、setup函数

```
void setup(){  
    item = false;  
    hz = 0;  
    pinMode(2, INPUT); //管脚2设置为输入模式  
    pinMode(3, INPUT);  
    //类外函数，参数为管脚号，软件中断函数，软件中断模式  
    PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,RISING);  
    PCintPort::attachInterrupt(3,attachPinInterrupt_fun_3,FALLING);  
    pinMode(5, OUTPUT);  
    pinMode(9, OUTPUT);  
}
```

6、loop函数

```
void loop(){  
    if (item) {  
        while (item) {  
            sound(); //调用sound函数  
            digitalWrite(9,!digitalRead(9)); //9号管脚写入读取到9号管脚电平相反值。  
            delay(10);  
        }  
    } else {  
        item = false;  
        noTone(5); //调用结束播放函数  
        digitalWrite(9,LOW); //9号管脚写入低电平  
    }  
}
```



项目简析



项目任务

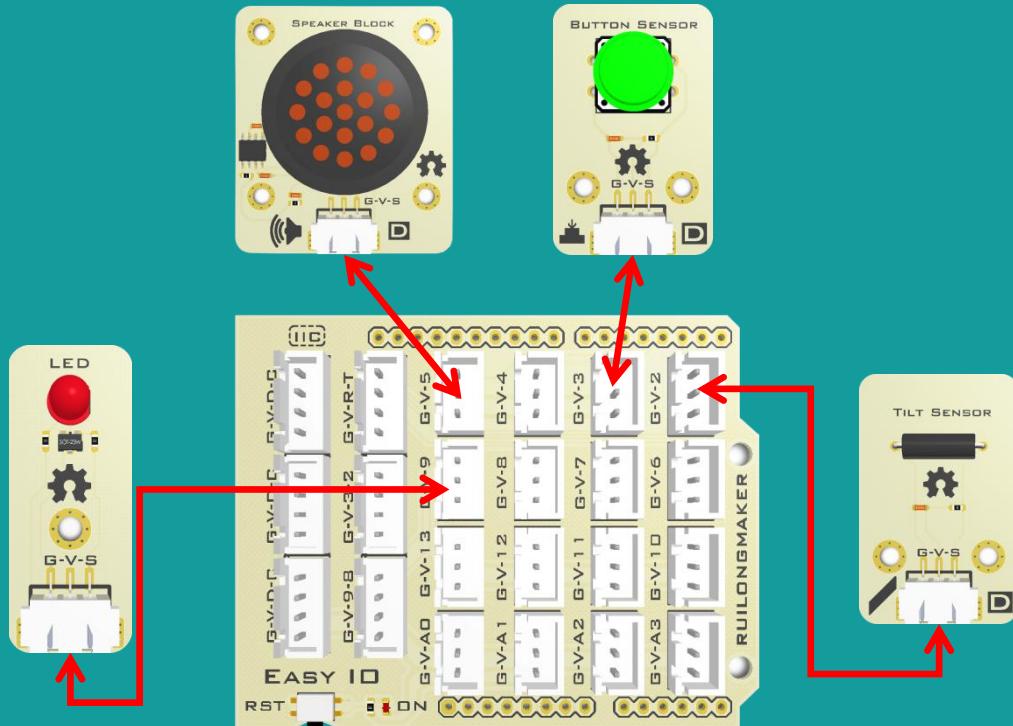
利用震动传感器、喇叭扬声器模块、按钮传感器、LED模块制作车辆防盗报警器，当产生震动时，发出警报，并且人为的按下按钮，报警才会停止、LED灯熄灭。

思路分析

震动报警器设置中断连接D2、按钮传感器设置中断连接D3，编写警报声函数，利用三角函数运算获取不同数值，来改变声音的频率，从而模拟警报声，当震动传感器检测到持续震动，喇叭扬声器模块发出警报，点亮LED灯，按下按钮，触发中断，结束警报声，LED灯熄灭。



硬件连线



序号	名称	连线	备注
1	震动传感器	D2	
2	按钮传感器	D3	
3	喇叭扬声器模块	D5	
4	LED模块	D9	



程序编程



- 1、初始化变量并赋值为0。
- 2、管脚2连接震动传感器，设置中断，震动传感器默认为低电平（0），中断模式选择上升，管脚3连接按钮传感器，设置中断，按钮传感器默认高电平(1),中断模式选择下降。
- 3、定义sound函数，利用一个三角函数运算获取不同数值，来改变声音的频率，从而模拟警报声
- 4、如果震动传感器检测到持续震动，发出警报声，点亮LED灯。
- 5、按下按钮，触发中断，结束播放警报声，熄灭LED灯。



1、头文件引入

```
#include <PinChangeInt.h>
```

2、定义变量

```
volatile long starttime;
volatile boolean item;
volatile int hz;
float period;
float pulse;
```

3、中断函数

```
void attachPinInterrupt_fun_2() {
    item = true;
}
void attachPinInterrupt_fun_3() {
    item = false;
}
```

4、扬声器函数

扬声器函数, int tonePin表示管脚号, int frequency表示声音频率, int duration表示持续时间

```
void newtone(int tonePin, int frequency, int duration)
{
    float period = 1000000.0 /frequency;
    float pulse = period / 2.0;
    for (int i=1; i<=((duration * 1000.0)/period);i=i+1)
    {
        pinMode(tonePin, OUTPUT);//管脚设置为输出模式
        digitalWrite(tonePin,HIGH);//向管脚写入高电平
        delayMicroseconds(pulse);//延时ms
        pinMode(tonePin, OUTPUT);
        digitalWrite(tonePin,LOW);//向管脚写入低电平
        delayMicroseconds(pulse);
    }
}
```

6、sound函数

```
void sound(){
    for (int i = 0; i <= 179; i = i + (1)) {
        hz = 2000 + round(sin(i / 180.0 * 3.14159) * 1000); //三角函数运算频率
        newtone(5,hz,10); //调用newtone函数, 传递实参。
    }
}
```

7、setup函数

```
void setup(){
    starttime = 0;
    item = false;
    hz = 0;
    pinMode(2, INPUT); //管脚2设置为输入模式
    pinMode(3, INPUT); //管脚3设置为输入模式
    //类外函数, 传递参数分别为管脚号, 中断函数、模式RISING代表上升, 电平从低电平到高电平
    PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,RISING);
    //类外函数, 传递参数分别为管脚号, 中断函数、模式RISING代表下降, 电平从高电平到低电平
    PCintPort::attachInterrupt(3,attachPinInterrupt_fun_3,FALLING);
    pinMode(5, OUTPUT); //管脚5设置为输出模式
    pinMode(9, OUTPUT);
}
```

8、loop函数

```
void loop(){
    if (item) {
        starttime = millis();
        while (item) {
            sound(); //调用sound函数
            digitalWrite(9,HIGH); //调用digitalwrite函数, 传递参数意义为向9号管脚写入高电平。
        }
    } else {
        item = false;
        digitalWrite(9,LOW);
    }
}
```



项目简析



项目任务

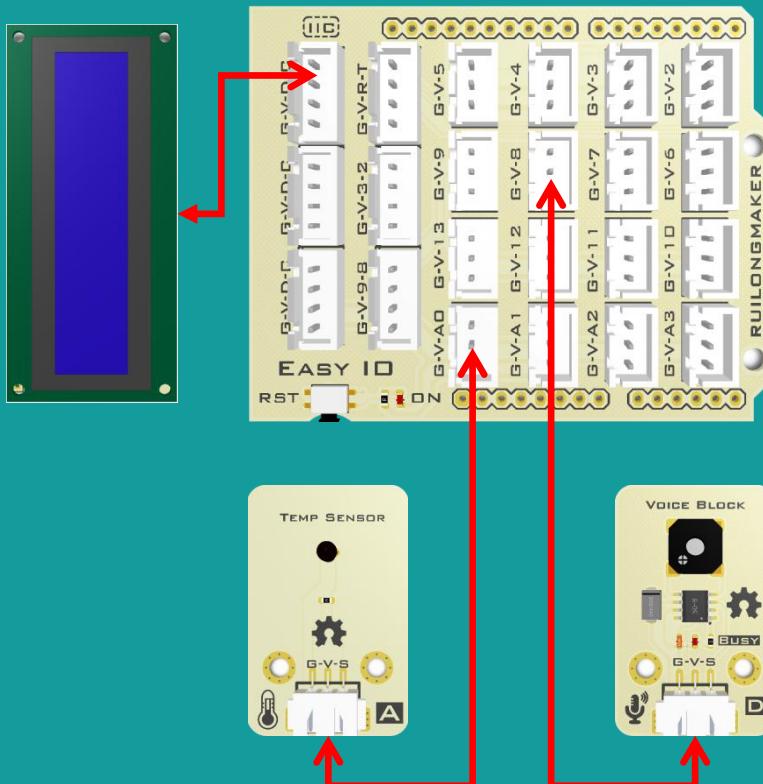
通过LM35温度传感器、LCD1602显示屏、语音模块，制作一个语音播报温度计。

思路分析

制作语音温度计，首先编写“broadcast”函数，当输入数字0-9时，播放数字，对LCD显示器显示当前温度，随后对温度进行十位个位分离语音模块按照正常逻辑播放。



硬件连线



序号	名称	连线	备注
1	LCD显示器	GVDC	
2	语音模块	D8	
3	LM35温度传感器	A0	



程序编程





1、头文件引入

```
#include<RL_Voice68.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

2、定义变量

```
volatile int temp;
```

3、创建对象

```
VOICE_68 voice_8(8);
LiquidCrystal_I2C mylcd(0x27,16,2);
```

4、播放函数

```
void broadcast(int num) {
    switch (num) {
        case 0:
            voice_8.send_data(0x2C); delay(500);break;//调用播放声音函数
        case 1:
            voice_8.send_data(0x23); delay(500);break;
        case 2:
            voice_8.send_data(0x24); delay(500);break;
        case 3:
            voice_8.send_data(0x25); delay(500);break;
        case 4:
            voice_8.send_data(0x26); delay(500); break;
        case 5:
            voice_8.send_data(0x27); delay(500);break;
        case 6:
            voice_8.send_data(0x28); delay(500);break;
        case 7:
            voice_8.send_data(0x29); delay(500);break;
        case 8:
            voice_8.send_data(0x2A); delay(500); break;
        case 9:
            voice_8.send_data(0x2B); delay(500);
            break;
    }
}
```

5、setup函数

```
void setup(){
    voice_8.begin();
    temp = 0;
    mylcd.init(); //显示屏初始化
    mylcd.backlight();
}
```

6、loop函数

```
void loop(){
    temp = analogRead(A0)*0.488; //读取管脚A0模拟值并且转化为摄氏温度值
    mylcd.setCursor(0, 0);
    mylcd.print("Hi..ruilongmaker");
    mylcd.setCursor(0, 1);
    mylcd.print("");
    mylcd.setCursor(1-1, 2-1);
    mylcd.print(String(String("temp:") + String(temp)) + String(".C"));
    voice_8.send_data(0x2D); //volume control 0xE0-E7;
    delay(500);
    voice_8.send_data(0x38); //volume control 0xE0-E7;
    delay(500);
    voice_8.send_data(0x34); //volume control 0xE0-E7;
    delay(500);
    if (temp / 10) {
        broadcast(temp / 10);
        voice_8.send_data(0x22); //volume control 0xE0-E7;
        delay(500);
    }
    if ((long) (temp) % (long) (10)) {
        broadcast((long) (temp) % (long) (10)); //调用broadcast函数，播放声音
    }
}
```



项目简析



项目任务

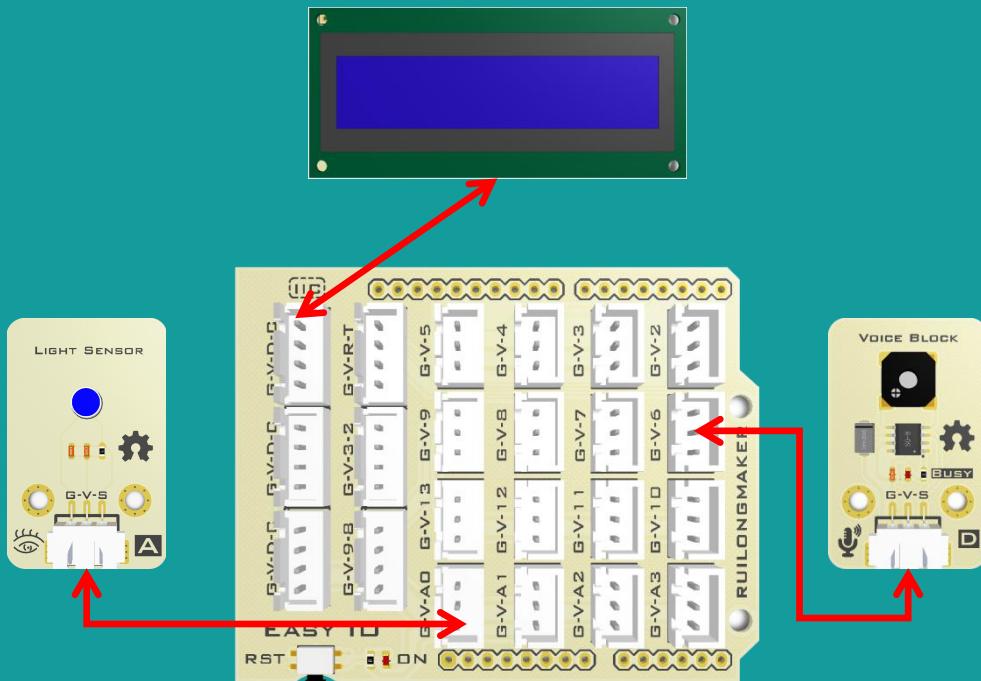
通过光线传感器、语音模块、LCD显示屏制作一个“语音光线强度计”，语音播报光线传感器获取到的模拟值，LCD显示当前光线强度。

思路分析

制作语音光线强度，首先编写“读数”函数，当输入数字0-9时，播放数字，对光线传感器获取到的光线强度模拟值进行千位、百位、十位、个位分离，显示在LCD显示屏上，调用读数函数，按照正常逻辑播放当前光线强度。



硬件连线



序号	名称	连线	备注
1	LCD显示器	GVDC	
2	语音模块	D6	
3	光线传感器	A0	



程序编程

编写读数函数



获取光线强度
模拟值

光线强度千位、
百位、十位、个
位分离

分离后的数据显
示在LCD上

调用“dushu”函
数，按照正常
逻辑播放。



1、头文件引入

```
#include<RL_Voice68.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

2、定义变量

```
volatile int guangxian;
volatile int qian;
volatile int bai;
volatile int shi;
volatile int ge;
```

3、创建对象

```
VOICE_68 voice_5(5);
```

4、读数函数编写

```
void dushu(int num) {switch (num) {
    case 0://参数num满足条件，跳出，不向下执行，否则继续向下执行。
        voice_4.send_data(0x2C);delay(500);break;
    case 1:
        voice_4.send_data(0x23);delay(500);break;
    case 2:
        voice_4.send_data(0x24);delay(500);break;
    case 3:
        voice_4.send_data(0x25);delay(500);break;
    case 4:
        voice_4.send_data(0x26);delay(500); break;
    case 5:
        voice_4.send_data(0x27);delay(500);break;
    case 6:
        voice_4.send_data(0x28);delay(500);break;
    case 7:
        voice_4.send_data(0x29);delay(500);break;
    case 8:
        voice_4.send_data(0x2A);delay(500);break;
    case 9:
        voice_4.send_data(0x2B);delay(500);break;}}
```

5、setup函数

```
void setup(){
    guangxian = 0;
    qian = 0;
    bai = 0;
    shi = 0;
    ge = 0;
    mylcd.init();
    mylcd.backlight();
    voice_5.begin();}
```

6、loop函数

```
void loop(){
    guangxian = analogRead(A0);
    mylcd.setCursor(0, 0);
    mylcd.print("Hi..ruilongmaker");
    mylcd.setCursor(0, 1);
    mylcd.print("");
    qian = (long) ((guangxian / 1000)) % (long) (10);
    bai = (long) ((guangxian / 100)) % (long) (10);
    shi = (long) ((guangxian / 10)) % (long) (10);
    ge = (long) ((guangxian / 1)) % (long) (10);
    mylcd.setCursor(1-1, 2-1);
    mylcd.print(String(String(String("light:") + String(qian)) + String(bai)) + String(shi)) +
    String(ge));
    voice_5.send_data(0x2D); //volume control 0xE0-E7;delay(500);
    voice_5.send_data(0x34); //volume control 0xE0-E7;delay(500);
    if (qian) {dushu(qian);
        voice_5.send_data(0x20); //volume control 0xE0-E7;delay(500);}
    if (bai) {dushu(bai);
        voice_5.send_data(0x21); //volume control 0xE0-E7;
        delay(500);}
    if (shi) {
        dushu(shi);
        voice_5.send_data(0x22); //volume control 0xE0-E7;
        delay(500);}
    if (ge) {dushu(ge);}}
```



项目简析



项目任务

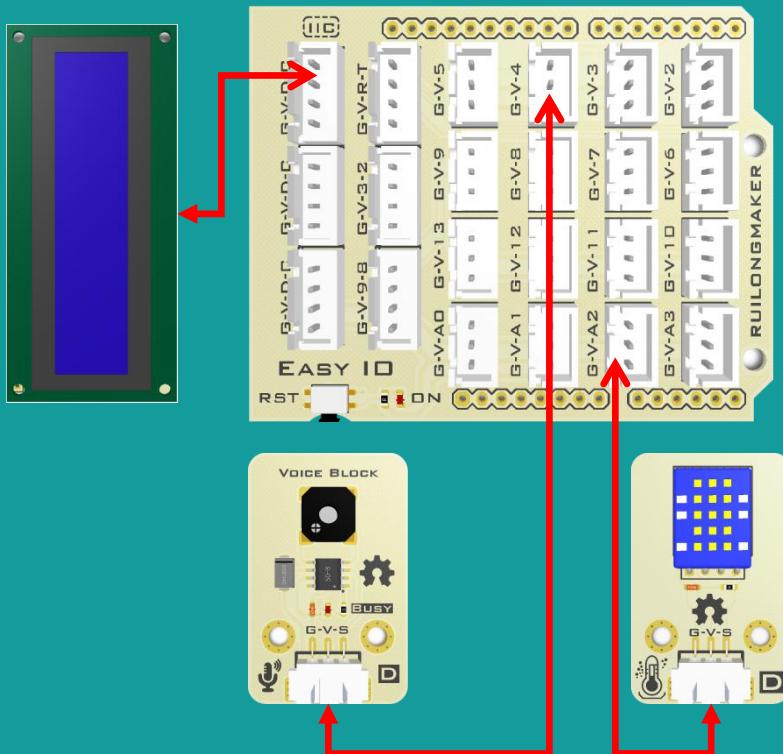
通过温湿度传感器，声音传感器和OLED显示器模块，实现温湿度的语音播报并且显示在显示器的屏幕上。

思路分析

制作智能语音温湿度计。需要一个温湿度传感器，声音传感器和LCD显示器模块。编写“dushu”函数，当输入数字0-9时，播放数字，对温湿度进行取整，LCD显示器显示当前温度湿度，随后对温湿度进行十位个位分离声音传感器按照正常逻辑播放。



硬件连线

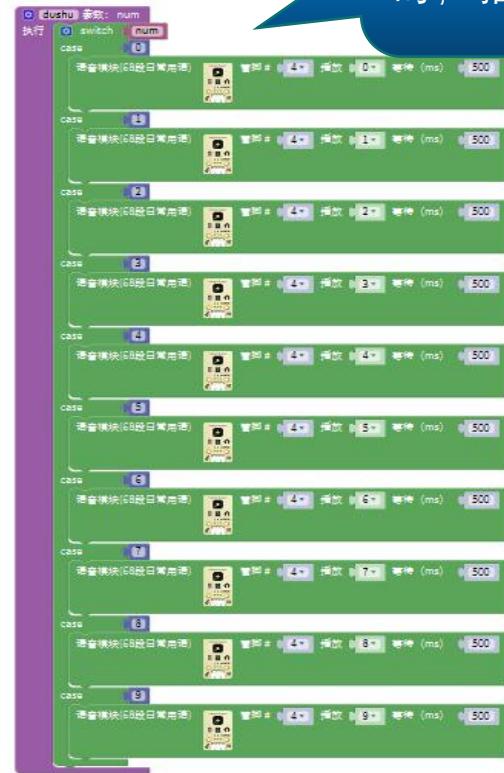


序号	名称	连线	备注
1	LCD显示器	GVDC	
2	语音模块	D4	
3	DTH11温湿度传感器	A2	



程序编程

编写“dushu”函数，
当输入0-9数字时，播放数字。



获取温度
湿度数值，
显示在
LCD上。

温度数值
个十位分
离，按照逻
辑播报温
度数值。



第一行，显示
Hi..ruilongmak
er,并播报一次
“当前”。

湿度数
值个十
位分离，
按照逻
辑播报
湿度数
值。





1、头文件引入

```
#include<RL_Voice68.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
```

2、定义变量

```
volatile int wendu; volatile int shidu;
volatile int wendu_shi; volatile int wendu_ge;
volatile int shidu_shi; volatile int shidu_ge;
```

3、创建对象

```
VOICE_68 voice_4(4);
LiquidCrystal_I2C mylcd(0x27,16,2);
```

4、读数函数编写

```
void dushu(int num) {switch (num) {
    case 0://参数num满足条件，跳出，不向下执行，否则继续向下执行。
        voice_4.send_data(0x2C);delay(500);break;
    case 1:
        voice_4.send_data(0x23);delay(500);break;
    case 2:
        voice_4.send_data(0x24);delay(500);break;
    case 3:
        voice_4.send_data(0x25);delay(500);break;
    case 4:
        voice_4.send_data(0x26);delay(500); break;
    case 5:
        voice_4.send_data(0x27);delay(500);break;
    case 6:
        voice_4.send_data(0x28);delay(500);break;
    case 7:
        voice_4.send_data(0x29);delay(500);break;
    case 8:
        voice_4.send_data(0x2A);delay(500);break;
    case 9:
        voice_4.send_data(0x2B);delay(500);
        break;}}
```

5、setup函数

```
void setup(){
    voice_4.begin();
    wendu = 0;
    shidu = 0;
    wendu_shi = 0;
    wendu_ge = 0;
    shidu_shi = 0;
    shidu_ge = 0;
    mylcd.init();
    mylcd.backlight();
    voice_4.send_data(0x2D); //volume control 0xE0-E7;
    delay(600);
    mylcd.setCursor(0, 0);
    mylcd.print("Hi.ruilongmaker");
    mylcd.setCursor(0, 1);
    mylcd.print("");
    dhtA2.begin();
}
```

6、loop函数

```
void loop(){
    wendu = dhtA2.readTemperature(); //获取温度
    shidu = dhtA2.readHumidity(); //获取湿度
    mylcd.setCursor(3-1, 2-1);
    mylcd.print(String(String(wendu, DEC)) + String(".C")); //显示温度
    mylcd.setCursor(11-1, 2-1);
    mylcd.print(String(String(shidu, DEC)) + String("%")); //湿度
    wendu_shi = wendu / 10;
    wendu_ge = (long) (wendu) % (long) (10); //温度十位个位分离
    voice_4.send_data(0x2D);
    delay(600);
    voice_4.send_data(0x38);
    delay(600);
    voice_4.send_data(0x34);
    delay(600);
```



```
if (wendu_shi) { //调用读数函数
    dushu(wendu_shi);
    voice_4.send_data(0x22);
    delay(600); }
if (wendu_ge) {
    dushu(wendu_ge);}
shidu_shi = shidu / 10;
shidu_ge = (long) (shidu) % (long) (10);
voice_4.send_data(0x39);
delay(600);
voice_4.send_data(0x34);
delay(600);
if (shidu_shi) {
    dushu(shidu_shi);
    voice_4.send_data(0x22);
    delay(600);}
if (shidu_ge) {
    dushu(shidu_ge);}
delay(1000);}
```



项目简析



项目任务

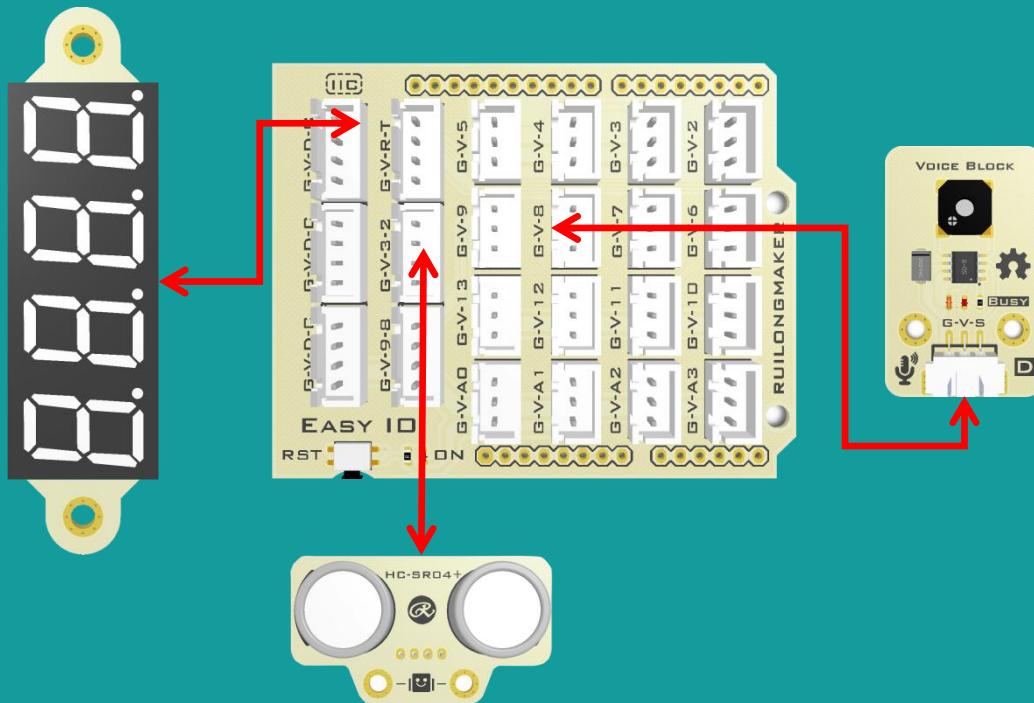
通过超声波传感器，声音传感器，数码管模块制作一个语音超声测距仪。

思路分析

制作作语音测距仪，编写“dushu”函数，当输入数字0-9时，播放数字，对超声波传感器获取到的数值进行取整，数码管模块显示数值，随后对获取到的数值进行分离，分别播报对应数值，如果当前位数字有“0”，越过播报。



硬件连线

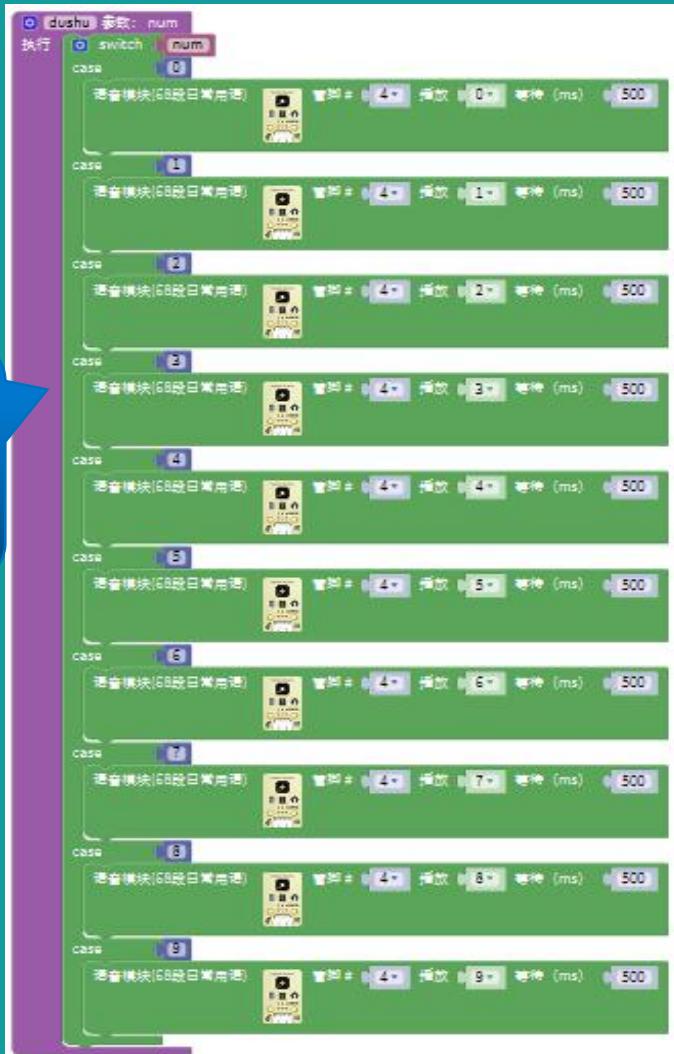


序号	名称	连线	备注
1	数码管模块	GVDC	
2	声音传感器	D8	
3	超声波传感器	GV32	



程序编程

编写“dushu”函数，
当输入0-9数字时，
播报数字。



获取超声波传
感器数值，并
取整。

数值显示在数
码管上。

分离四位数值，
千，百，十，
个。

分别播报对应
数值的值。如
果该位是“0”，
越过播报。



1、头文件引入

```
#include<RL_Voice68.h>
VOICE_68 voice_8(8);
#include <Wire.h>
#include <TM1650.h>
```

2、定义变量

```
volatile int item;
volatile int qian;
volatile int bai;
volatile int shi;
volatile int ge;
```

3、读数函数编写

```
void dushu(int num) {switch (num) {
    case 0://结构体，参数num满足条件，跳出，不向下执行，否则继续向下执行。
        voice_4.send_data(0x2C);delay(500);break;
    case 1:
        voice_4.send_data(0x23);delay(500);break;
    case 2:
        voice_4.send_data(0x24);delay(500);break;
    case 3:
        voice_4.send_data(0x25);delay(500);break;
    case 4:
        voice_4.send_data(0x26);delay(500); break;
    case 5:
        voice_4.send_data(0x27);delay(500);break;
    case 6:
        voice_4.send_data(0x28);delay(500);break;
    case 7:
        voice_4.send_data(0x29);delay(500);break;
    case 8:
        voice_4.send_data(0x2A);delay(500);break;
    case 9:
        voice_4.send_data(0x2B);delay(500);
        break;}}
```

4、checkdistance_3_2函数

```
float checkdistance_3_2() {
    digitalWrite(3, LOW);
    delayMicroseconds(2);
    digitalWrite(3, HIGH);
    delayMicroseconds(10);
    digitalWrite(3, LOW);
    float distance = pulseIn(2, HIGH) / 58.00;
    delay(10);
    return distance;
}
```

5、setup函数

```
void setup(){
    voice_8.begin();
    item = 0;
    qian = 0;
    bai = 0;
    shi = 0;
    ge = 0;
    pinMode(3, OUTPUT);
    pinMode(2, INPUT);
    Wire.begin();
    ruilong_4display.init();
}
```



6、loop函数

```
void loop(){
    item = round(checkdistance_3_2());
    rui long_4display.displayString(item);
    qian = item / 1000;
    bai = (long) ((item / 100)) % (long) (10);
    shi = ((long) (item) % (long) (100)) / 10;
    ge = (long) (((long) (item) % (long) (100))) % (long) (10);
    if (qian) {
        dushu(qian);
        voice_8.send_data(0x20); //volume control 0xE0-E7;
        delay(500);

    }
    if (bai) {
        dushu(bai);
        voice_8.send_data(0x21); //volume control 0xE0-E7;
        delay(500);

    }
    if (shi) {
        dushu(shi);
        voice_8.send_data(0x22); //volume control 0xE0-E7;
        delay(500);

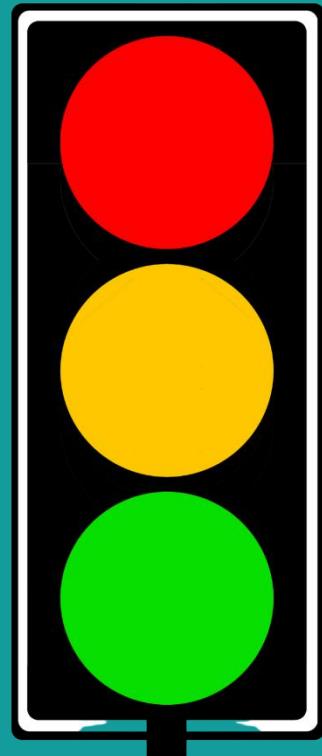
    }
    if (ge) {
        dushu(ge);

    }
    voice_8.send_data(0x0F); //volume control 0xE0-E7;
    delay(1000);
    delay(1000);

}
```



项目简析



项目任务

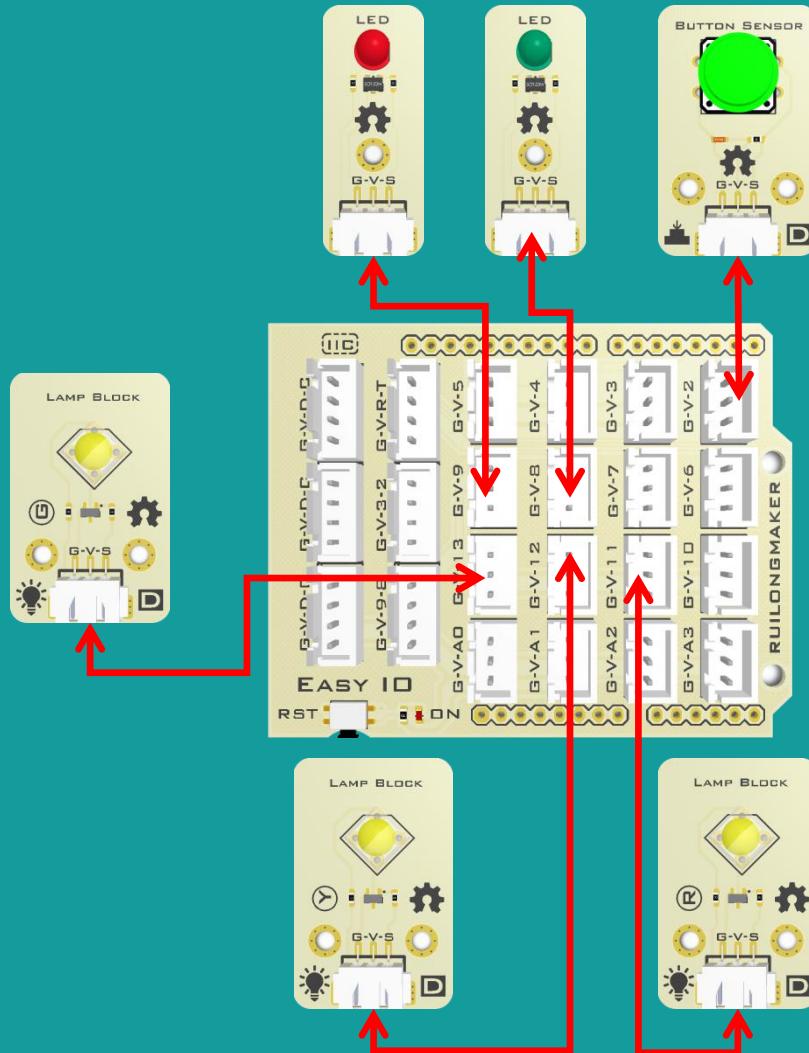
通过按钮传感器、闪灯模块（红、黄、绿）、LED模块（红、绿）制作一个简易交通灯。

思路分析

在有些城市安装了“人行过街按钮”，在一般情况下，机动车道上一直亮着绿灯，斑马线上亮红灯，当有人想要过马路，按下按钮，机动车道会在一段时间后亮起红灯，斑马线上亮起绿灯，让行人安全过马路。



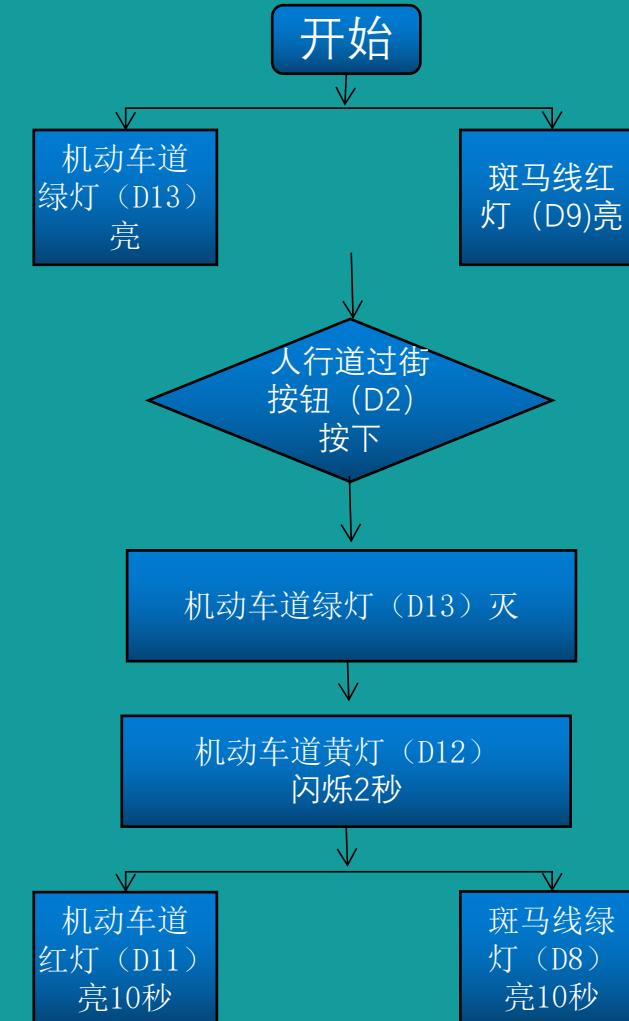
硬件连线



序号	名称	连线	备注
1	按钮传感器	D2	切换红绿灯
2	闪灯模块（红）	D11	用作机动车道红灯
3	闪灯模块（黄）	D12	用作机动车道黄灯
4	闪灯模块（绿）	D13	用作机动车道绿灯
5	LED模块（绿）	D8	用作斑马线绿灯
6	LED模块（红）	D9	用作斑马线红灯



程序编程





1、setup函数

```
void setup(){  
    /*初始化引脚配置为OUTPUT模式  
    低阻抗状态，可以用于点亮LED灯。  
    */  
    pinMode(13, OUTPUT);  
    pinMode(12, OUTPUT);  
    pinMode(11, OUTPUT);  
    pinMode(9, OUTPUT);  
    pinMode(8, OUTPUT);  
    pinMode(2, INPUT);  
}
```

2、loop函数

```
void loop(){  
    //设置LED灯亮灭状态  
    digitalWrite(13,HIGH);  
    digitalWrite(12,LOW);  
    digitalWrite(11,LOW);  
    digitalWrite(9,HIGH);  
    digitalWrite(8,LOW);  
    /*按钮传感器默认为高电平（1）  
    按下为低电平（0），对电平进行取反*/  
    if (!digitalRead(2)) {  
        //十三号管脚所连LED灯熄灭  
        digitalWrite(13,LOW);  
        //黄灯每隔两秒熄灭一次，循环五次  
        for (int i = 1; i <= 5; i = i + (1)) {  
            digitalWrite(12,HIGH);  
            delay(2000);  
            digitalWrite(12,LOW);  
            delay(2000); }  
        //更改LED灯亮灭状态  
        digitalWrite(11,HIGH);  
        digitalWrite(9,LOW);  
        digitalWrite(8,HIGH);  
        digitalWrite(13,LOW);  
        delay(10000); } }
```



项目简析



项目任务

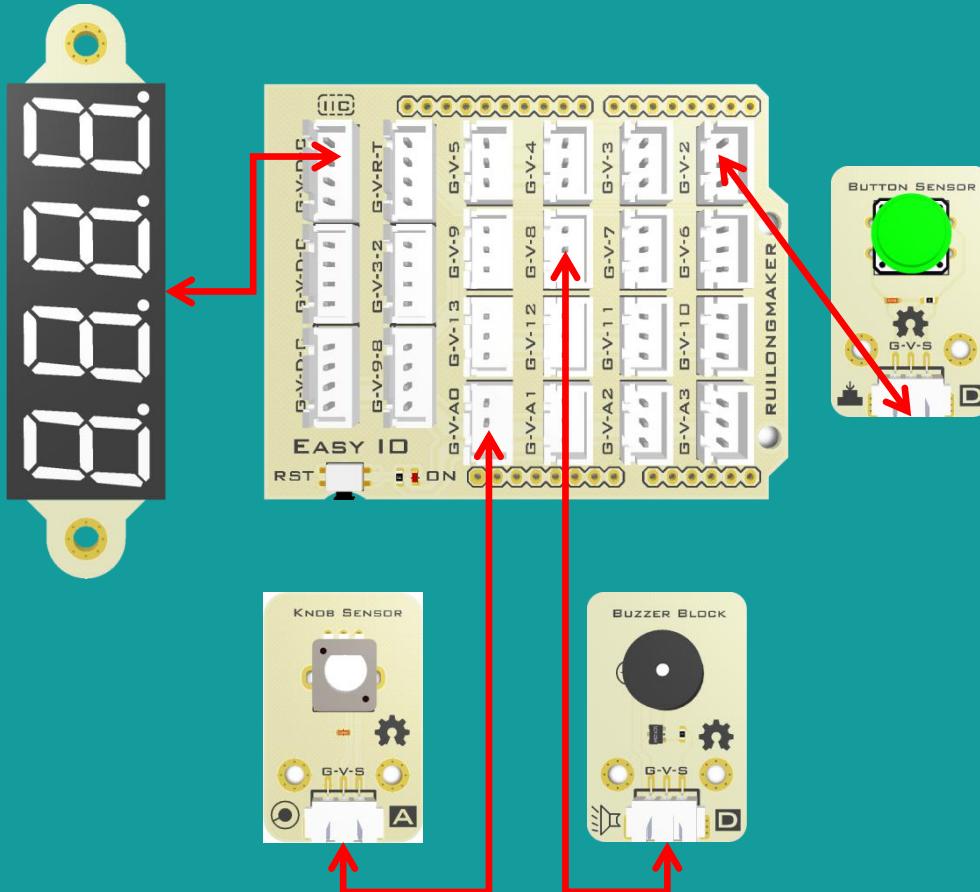
通过旋钮传感器、数码管模块、蜂鸣器模块、按钮传感器制作一个简易倒计时器。

思路分析

用中断管脚切换倒计时状态，通过旋钮设置倒计时秒数，程序设置为1-99秒，通过100倍关系将模拟输入管脚A0的数值映射为100-9900，通过闪屏显示数值，按下中断管脚所连接按钮传感器，倒计时开始，至倒计时结束后，闪屏显示0000，并伴随十次蜂鸣器蜂鸣。



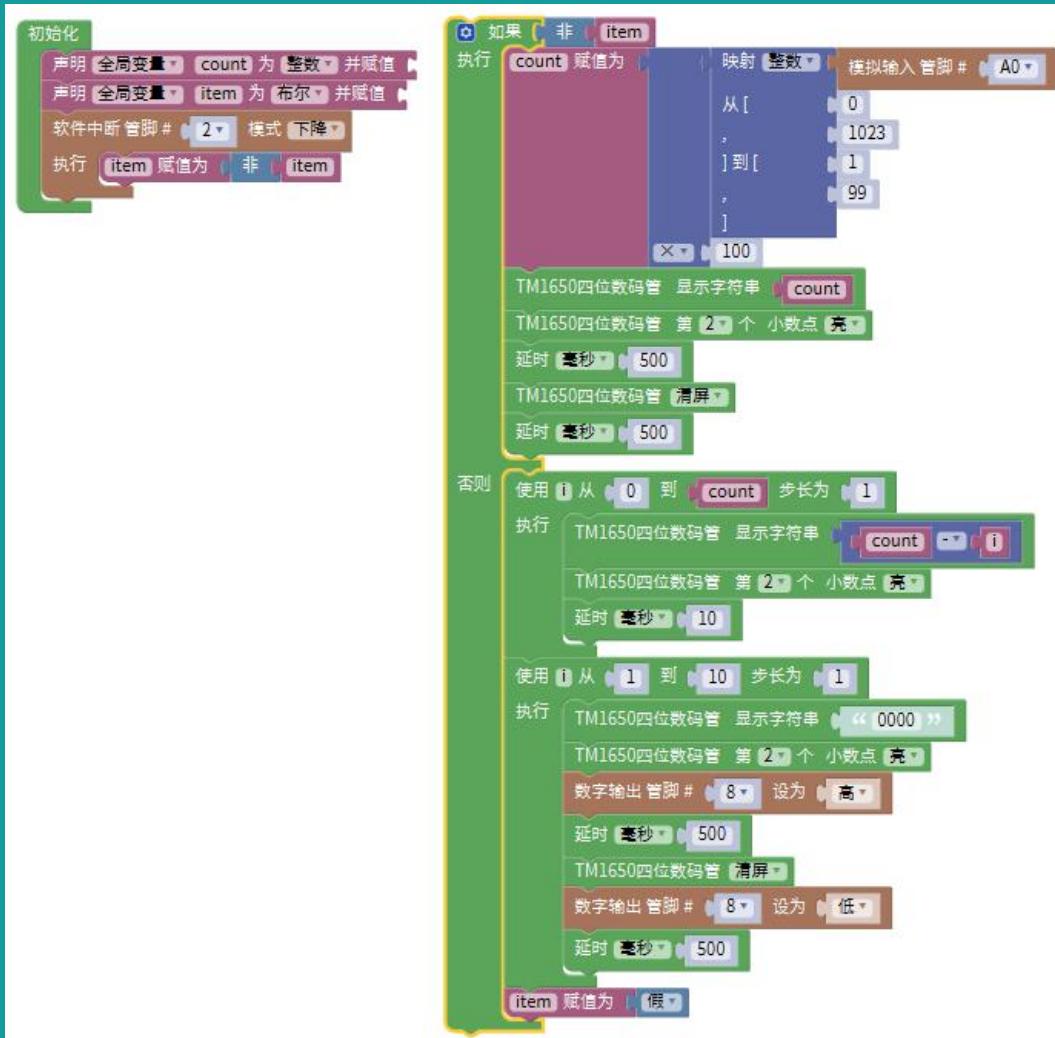
硬件连线



序号	名称	连线	备注
1	数码管模块	GVDC	
2	旋钮传感器	A0	
3	蜂鸣器模块	D8	
4	按钮传感器	D2	



程序编程



- 1、初始化中断管脚D2，用于切换设置状态及倒计时状态。
- 2、默认状态下通过旋钮设置1-99秒，通过100倍关系将模拟输入管脚A0的数值映射为100-9900，通过闪屏显示数值。
- 3、倒计时显示设定的100-9900范围数值，期间延时10毫秒的意义为循环一次，数值减少10毫秒，倒计时结束后，闪屏显示0000，并伴随十次蜂鸣器蜂鸣。



1、头文件引入

```
#include <PinChangeInt.h>
#include <Wire.h>
#include <TM1650.h>
```

2、定义变量

```
volatile int count;
volatile boolean item;
```

3、中断触发函数编写

```
void attachPinInterrupt_fun_2(){
    item = !item;
}
```

4、setup函数

```
void setup(){
    count = 0;
    item = 0;
    pinMode(2, INPUT);
/*
有强大的中断功能，使你可以在任何可获得的端口定义中断功能。你可以使用 attachInterrupt()函数直接指定引脚号
形式:attachInterrupt(interrupt, ISR, mode)
pin:中断引脚
ISR: 中断发生时的中断服务程序。这个函数必须没有参数没有返回值;
mode:定义中断触发类型，有四种形式:
    LOW: 低电平触发;
    CHANGE: 电平变化触发;
    RISING : 上升沿触发（由LOW变为HIGH）;
    FALLING: 下降沿触发（由HIGH变为LOW）;
*/
    PCintPort::attachInterrupt(2,attachPinInterrupt_fun_2,FALLING);
    Wire.begin();
//初始化屏幕
    ruilong_4display.init();
    pinMode(8, OUTPUT);
}
```

5、loop函数

```
void loop(){
    if (!item) {
        //获取到的count值取值范围为0-1023， 映射到100-9900
        count = (map(analogRead(A0), 0, 1023, 1, 99)) * 100;
        //数码管显示映射过后的数值
        ruilong_4display.displayString(count);
        //显示第二个小数点，从0开始
        ruilong_4display.setDot(1,true);
        delay(500);
        //清屏
        ruilong_4display.clear();
        delay(500);

    } else {
        //每隔十毫秒， count减少1.
        for (int i = (0); i <= (count); i = i + (1)) {
            ruilong_4display.displayString((count - i));
            ruilong_4display.setDot(1,true);
            delay(10);
        }
        //当count为0时， 数码管显示0000,
        for (int i = 1; i <= 10; i = i + (1)) {
            ruilong_4display.displayString("0000");
            ruilong_4display.setDot(1,true);
        }
        //8号管脚设置为高电平， 间隔为500毫秒
        digitalWrite(8,HIGH);
        delay(500);
        ruilong_4display.clear();
        digitalWrite(8,LOW);
        delay(500);
    }
    item = false;
}
```



项目简析



项目任务

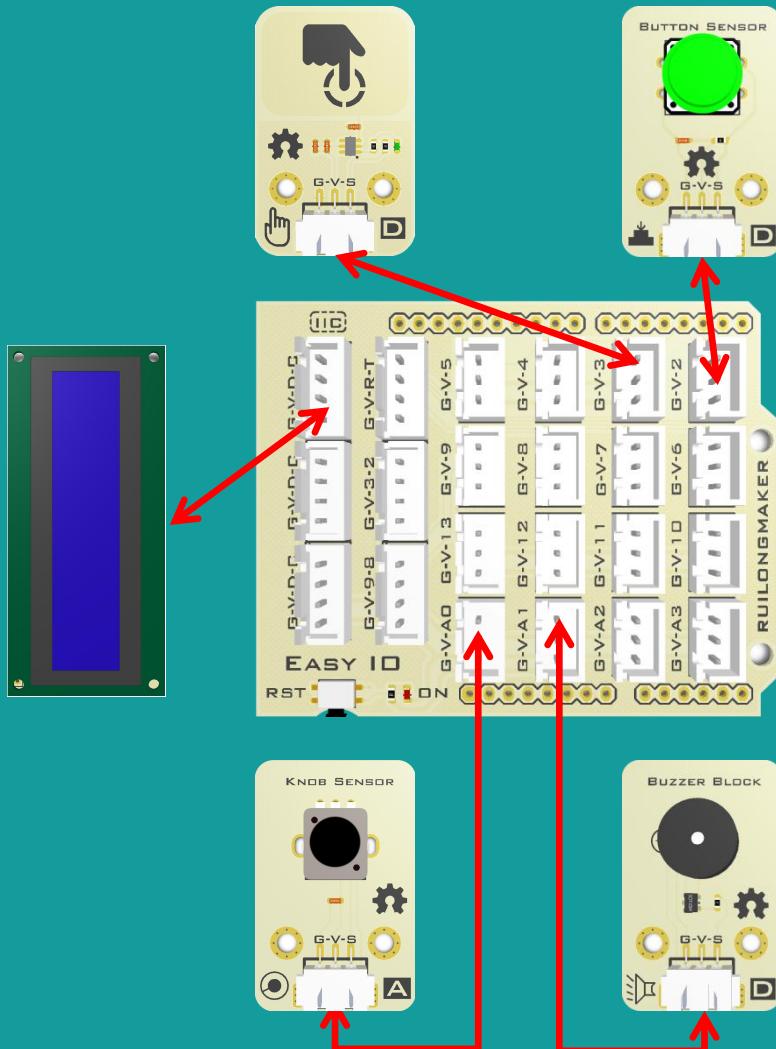
通过旋钮传感器、按钮传感器、LCD显示屏，触摸传感器、蜂鸣器模块，制作一个“触摸倒计时器”，通过旋钮传感器设置倒计时时间，刻度为分钟，按下按钮传感器，开始倒计时，倒计时结束、蜂鸣器十次蜂鸣、倒计时过程中，按下触摸传感器，倒计时停止，重新设置倒计时时间。

思路分析

触摸传感器连接D3管脚并设置软件中断，中断模式为电平上升，旋钮传感器是模拟传感器、建立旋钮传感器模拟值和倒计时时间的映射关系，用于设置倒计时时间，时间进行个位十位分离，显示在LCD显示屏上、当按下按钮传感器、开始倒计时、倒计时结束、蜂鸣器十次蜂鸣、倒计时过程中、按下触摸传感器、倒计时中断。



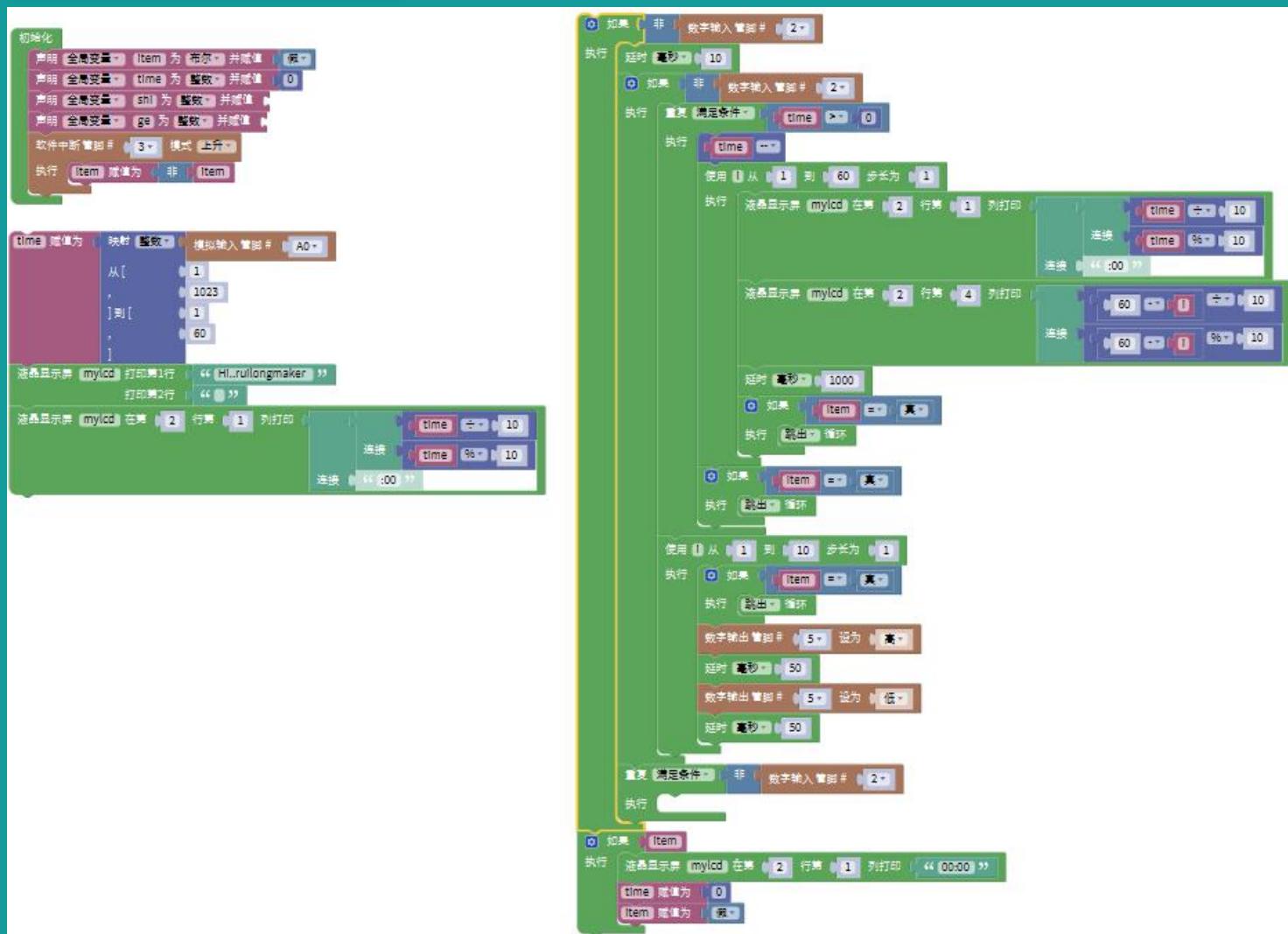
硬件连线



序号	名称	连线	备注
1	按钮传感器	D2	
2	触摸传感器	D3	
3	旋钮传感器	A0	
4	蜂鸣器模块	A1	
5	LCD显示屏	GVDC	



程序编程



- 1、初始化变量。
- 2、管脚3设置为软件中断。
- 3、建立旋钮传感器模拟值和倒计时时间的映射关系，倒计时时间最长设置为60分钟。
- 4、分离倒计时时间的十位、个位，并且显示在LCD上。
- 5、当连接2号管脚的按钮传感器被按下，开始倒计时。
- 6、倒计时过程中，按下触摸传感器，跳出循环，倒计时中断。
- 7、如果倒计时过程中、没有按下触摸传感器、倒计时结束后，蜂鸣器伴随十次蜂鸣。



1、头文件引入

```
#include <PinChangeInt.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

2、定义变量

```
volatile boolean item;
volatile int time;
volatile int shi;
volatile int ge;
```

3、中断触发函数编写

```
void attachPinInterrupt_fun_3() {
    item = !item;}
```

4、创建对象

```
LiquidCrystal_I2C mylcd(0x27,16,2); //创建含有三个实参的对象
```

5、setup函数

```
void setup(){
    item = false;
    time = 0;
    shi = 0;
    ge = 0;
    pinMode(3, INPUT);
    PCintPort::attachInterrupt(3,attachPinInterrupt_fun_3,RISING);
    pinMode(2, INPUT);
    pinMode(5, OUTPUT);
    mylcd.init();
    mylcd.backlight();
}
```

6、loop函数

```
void loop(){
    if (!digitalRead(2)) {
        delay(10);
        if (!digitalRead(2)) {
            while (time > 0) {
                time--;
                for (int i = 1; i <= 60; i = i + (1)) {
                    mylcd.setCursor(1-1, 2-1);
                    mylcd.print(String((time / 10)) + String(((long) (time) % (long) (10))) + String(":00"));
                    mylcd.setCursor(4-1, 2-1);
                    mylcd.print(String(((60 - i) / 10)) + String(((long) ((60 - i)) % (long) (10))));
                    delay(1000);
                    if (item == true) { break; }
                    if (item == true) {break;}
                }
                for (int i = 1; i <= 10; i = i + (1)) {
                    if (item == true) {
                        break;
                    }
                    digitalWrite(5,HIGH);
                    delay(50);
                    digitalWrite(5,LOW);
                    delay(50);}
                while (!digitalRead(2)) {}}
                if (item) {
                    mylcd.setCursor(1-1, 2-1);
                    mylcd.print("00:00");
                    time = 0;
                    item = false;}
                time = (map(analogRead(A0), 1, 1023, 1, 60));
                mylcd.setCursor(0, 0);
                mylcd.print("Hi..ruilongmaker");
                mylcd.setCursor(0, 1);
                mylcd.print("");
                mylcd.setCursor(1-1, 2-1);
                mylcd.print(String((time / 10)) + String(((long) (time) % (long) (10))) + String(":00"));}
```



项目简析



项目任务

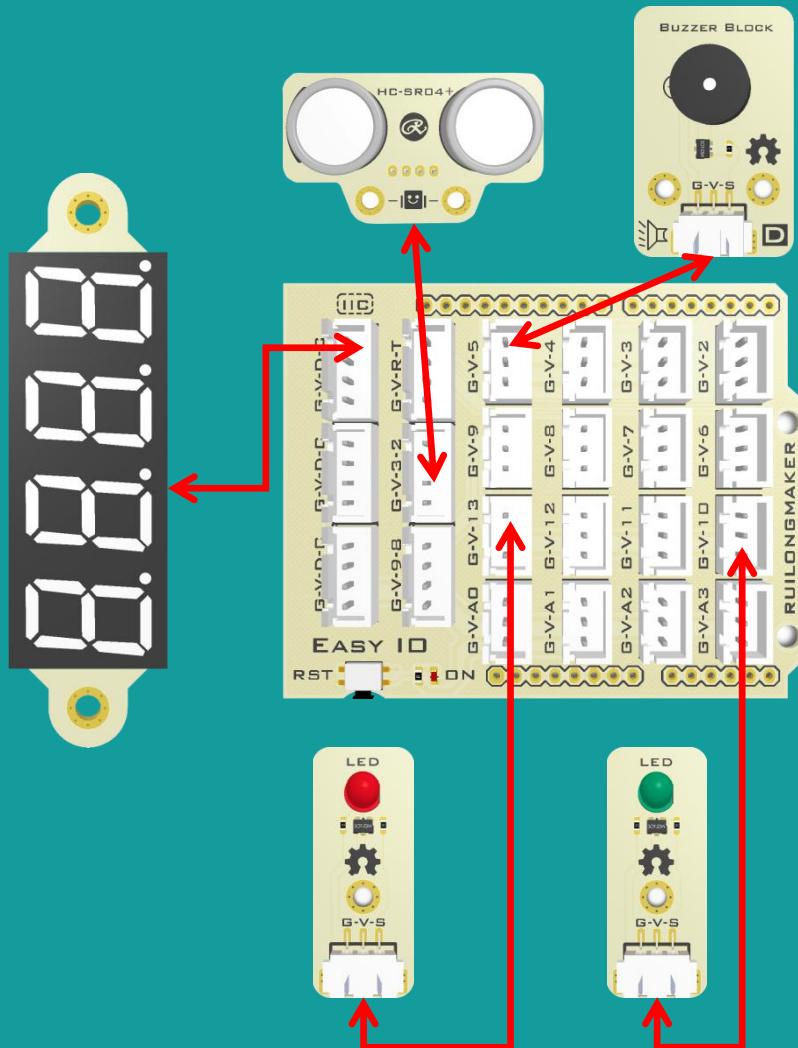
通过超声波传感器、数码管模块、LED模块
(红色、绿色、蜂鸣器模块制作一个智能坐姿
提醒警示器。)

思路分析

首先定义变量用于保存超声波传感器检测到的数值，并且通过数码管进行显示，如果检测到的数值小于30，蜂鸣器进行报警，红色LED灯亮起，示警调整坐姿，当数值大于30时，蜂鸣器停止蜂鸣，绿色LED灯亮起。



硬件连线



序号	名称	连线	备注
1	数码管模块	GVDC	
2	蜂鸣器模块	D5	
3	LED灯 (绿色)	D10	
4	LED灯 (红色)	D13	
5	超声波传感器	GV32	



程序编程



- 1、声明全局变量distance，用于保存超声波传感器数值。
- 2、数码管显示距离。
- 3、如果距离小于30，蜂鸣器蜂鸣，红色LED灯亮起，否则绿灯亮起，蜂鸣器待机状态。



1、头文件引入

```
#include <Wire.h>
#include <TM1650.h>
```

2、定义变量

```
volatile int distance;//用于保存超声波传感器数值
```

3、超声波传感器测距函数

```
float checkdistance_3_2() {
    digitalWrite(3, LOW); //3号管脚写入低电平
    delayMicroseconds(2); //延时2微秒 (us)
    digitalWrite(3, HIGH); //3号管脚写入高电平
    delayMicroseconds(10); //延时10微秒
    digitalWrite(3, LOW); //3号管脚写入低电平
/*
pulseIn函数是一个简单的测量脉冲宽度的函数，默认单位是us,pulseIn测出来的是超声波从发射到
接收所经过的时间,声速大约为343米/秒，即29.15us/cm,发送后接收到回波，要除以58。
*/
    float distance = pulseIn(2, HIGH) / 58.00;
    delay(10);
    return distance;
}
```

4、setup函数

```
void setup(){
//初始化变量
distance = 0;
Wire.begin();
//初始化数码管
ruilong_4display.init();
//pinMode()函数配置引脚
pinMode(3, OUTPUT); //管脚3为输出模式，低阻抗状态。
pinMode(2, INPUT); //管脚2为输入模式，高阻抗状态。
pinMode(5, OUTPUT);
pinMode(10, OUTPUT);
pinMode(13, OUTPUT);
}
```

5、loop函数

```
void loop(){
//数码管清屏
ruilong_4display.clear();
//超声波传感器测距函数返回值赋值给distance
distance = checkdistance_3_2();
//数码管显示distance
ruilong_4display.displayString(distance);
delay(100);
//如果distance小于30，蜂鸣器蜂鸣，红色LED亮，绿色LED灯灭。
if (distance < 30) {
    digitalWrite(5,HIGH);
    digitalWrite(10,LOW);
    digitalWrite(13,HIGH);
} else {
    digitalWrite(5,LOW);
    digitalWrite(10,HIGH);
    digitalWrite(13,LOW);
}
}
```



项目简析



项目任务

通过红外接收传感器、超声波传感器、马达风扇模块、红外软硅胶遥控器，LED模块、制作一个红外遥控感应风扇。

思路分析

开机上电默认为距离感应模式，绿色闪灯模块亮起，超声波传感器检测到障碍物的距离和马达风扇模块速度建立数学映射关系，距离越远，风扇速度越快，打开串口监视器，按下“R”时，十六进制编码为ffa857,按下“1”时为ff30af,按下“2”时为ff18e7,按下“3”时为ff7a85。遥控器“R”键用于切换模式，当处于遥控模式时，绿色闪灯模块熄灭，按下遥控器1、2、3可实现单独档位速度控制。



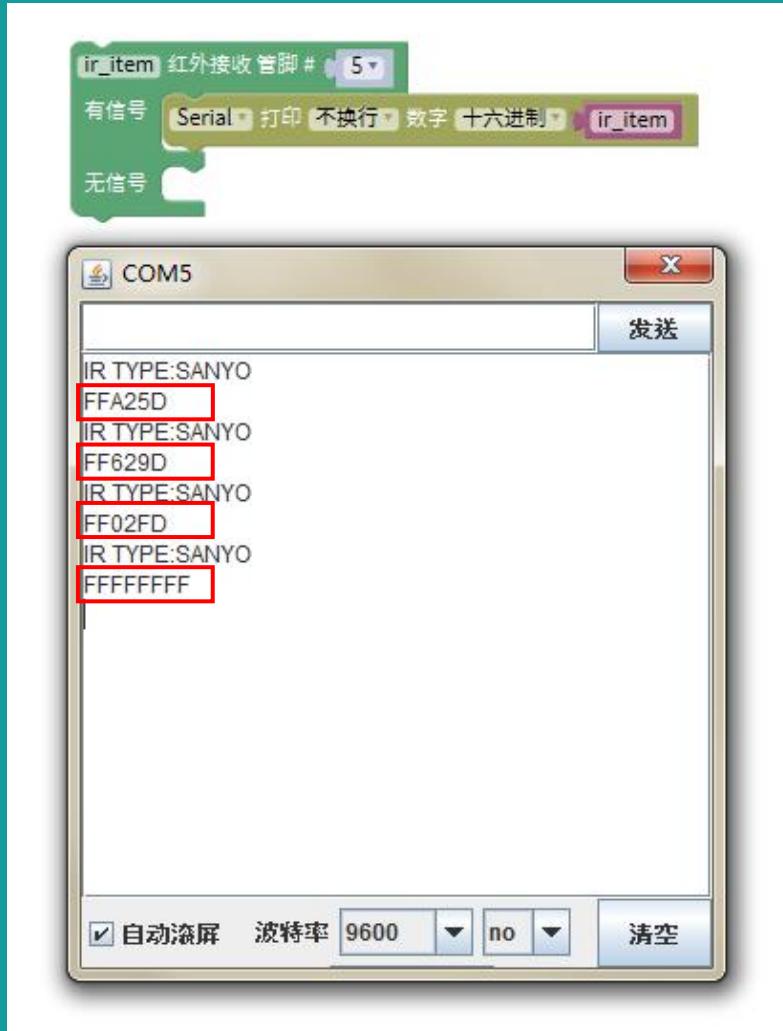
知识接入



- 1、在“通信”类模块中可以找到红外接收模块，该模块专门用于接收红外发射模块发出的红外信号。
- 2、Arduino主控板上D12管脚已经集成了一个红外接收传感器。
- 3、本案例外接了一个红外传感器，连接在D5管脚上。
- 4、本案例配套了一个红外软硅胶遥控器。



知识接入



1、上传程序后，打开串口监视器，用遥控器对准红外接收传感器，按下任意按键，可以看到一个十六进制的编码。每个按键对应不同的编码，按住某个按键不放，会发送一个重复编码“FFFFFFFF”

2、十六进制由0-9，A-F共十六个字符组成，遵循“十六进一”规则。规定在十六进制的数值前面加上前缀0x。



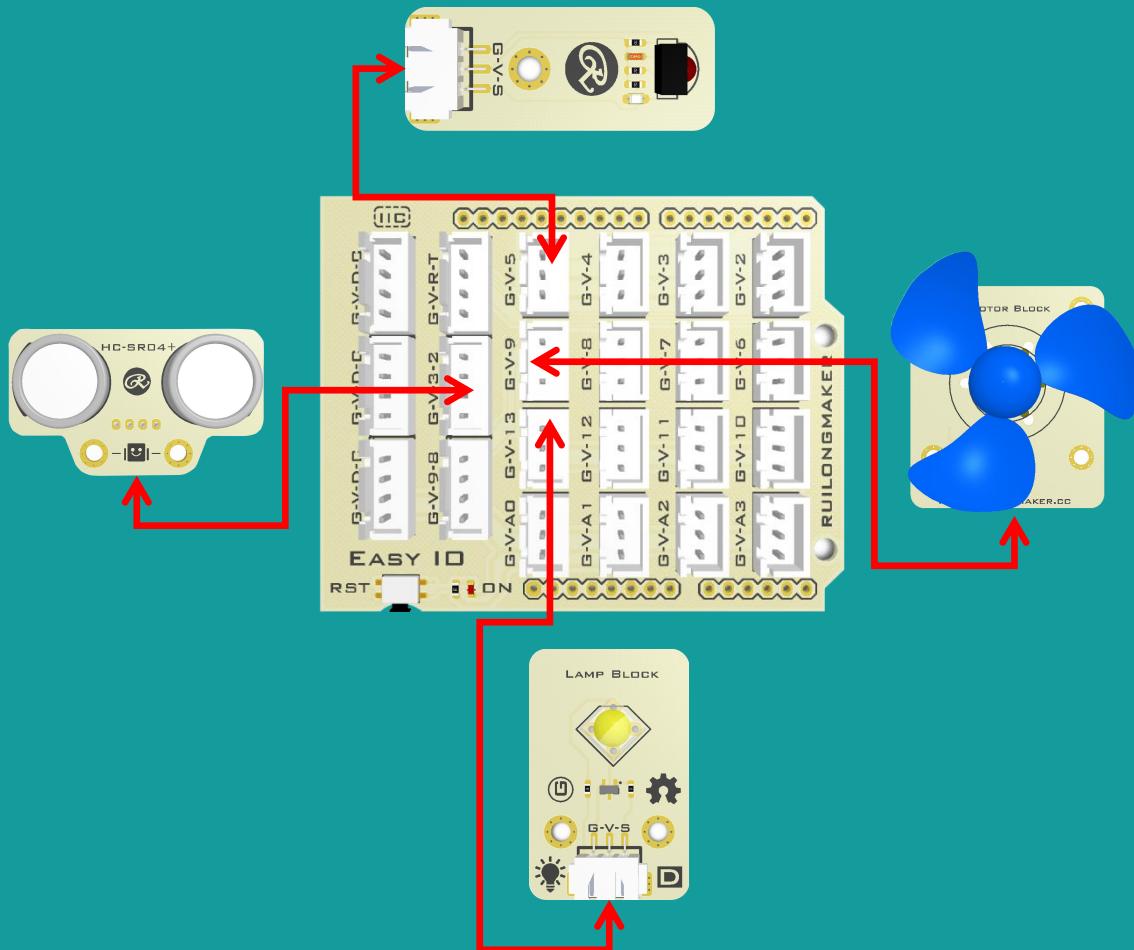
参考表格



键值	数值(H)	键值	数值(H)
A	FFA25D	0	FF6897
B	FF629D	1	FF30CF
C	FFE21D	2	FF18E7
D	FF22DD	3	FF7A85
E	FFC23D	4	FF10EF
F	FFB04F	5	FF38C7
上	FF02FD	6	FF5AA5
下	FF9867	7	FF42BD
左	FFE01F	8	FF4AB5
右	FF906F	9	FF52AD
中间R	FFA857		



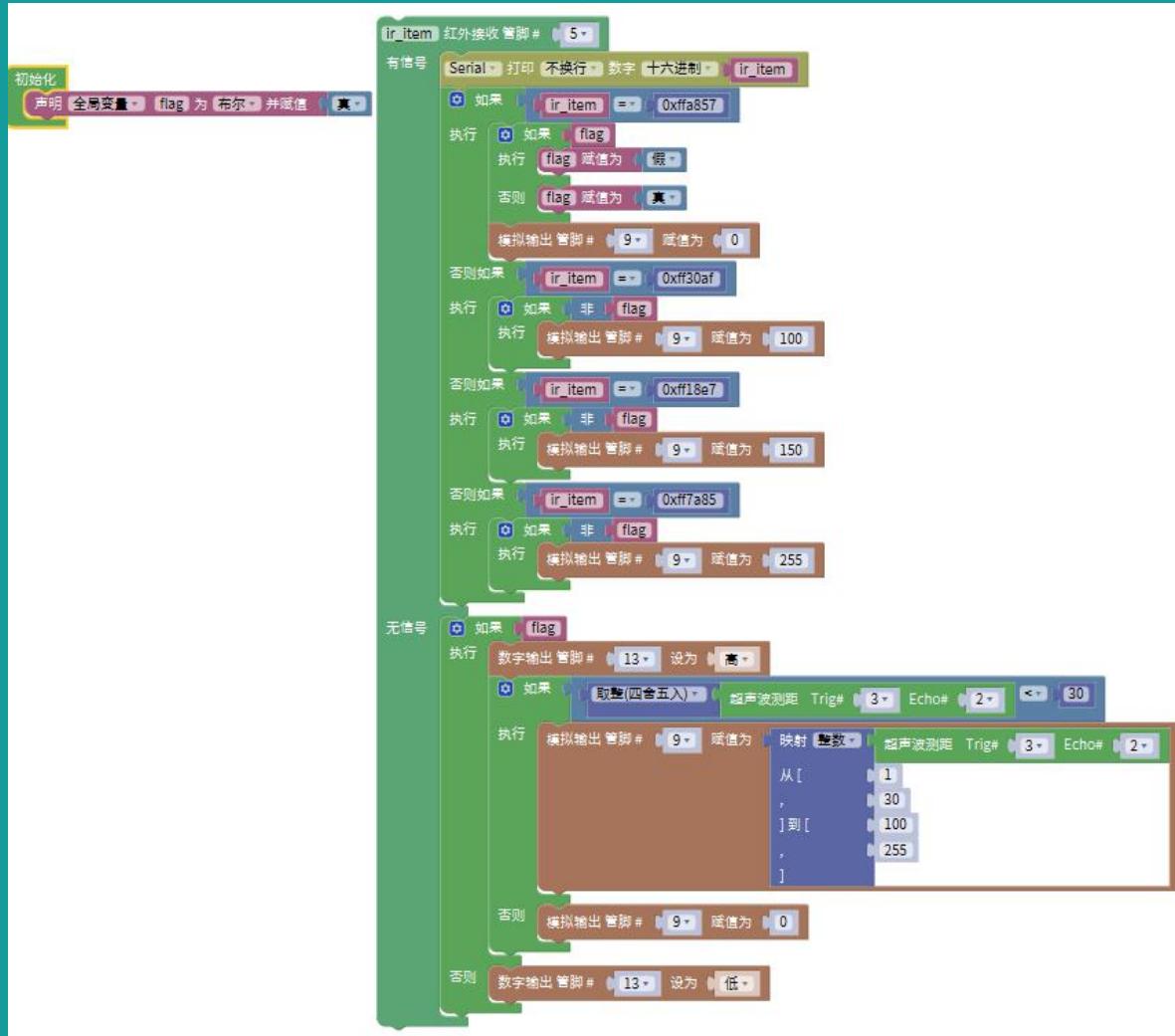
硬件连线



序号	名称	连线	备注
1	超声波传感器	GV32	
2	红外接收传感器	D5	
3	闪灯模块 (绿色)	D13	
4	马达风扇模块	D9	



程序编程



1、开机上电，默认为距离感应模式，绿色闪灯模块亮，当按下遥控器“R”时，切换模式，当处于遥控模式时，按下遥控器1、2、3可实现单独档位速度控制。

2、打开串口监视器，当按下“R”时，十六进制编码为ffa857,按下“1”时为ff30af,按下“2”时为ff18e7,按下“3”时为ff7a85。

3、距离感应模式下，绿色闪灯模块亮起，当距离值小于30时，距离值和电机速度二者关联映射，距离越远，电机转动速度越大，风速越大。



1、头文件引入

```
#include <IRremote.h>
```

2、定义变量

```
long ir_item;
IRrecv irrecv_5(5);
decode_results results_5;//结构体
volatile boolean flag;
```

3、超声波传感器测距函数

```
float checkdistance_3_2() {
    digitalWrite(3, LOW);//3号管脚写入低电平
    delayMicroseconds(2);//延时2微秒 (us)
    digitalWrite(3, HIGH);//3号管脚写入高电平
    delayMicroseconds(10);//延时10微秒
    digitalWrite(3, LOW);//3号管脚写入低电平
/*
```

pulseIn函数是一个简单的测量脉冲宽度的函数，默认单位是us,pulseIn测出来的是超声波从发射到接收所经过的时间,声速大约为343米/秒，即29.15us/cm,发送后接收到回波，要除以58。*/

```
    float distance = pulseIn(2, HIGH) / 58.00;
    delay(10);
    return distance;
}
```

4、setup函数

```
void setup(){
//设置串口波特率
Serial.begin(9600);
//管脚13为输出模式，低阻抗状态
pinMode(13, OUTPUT);
pinMode(3, OUTPUT);
//管脚2为输入模式，高阻抗状态
pinMode(2, INPUT);
//接收红外信号
irrecv_5.enableIRIn();
flag = true;
}
```

5、loop函数

```
void loop(){
//接收到信号
if (irrecv_5.decode(&results_5)) {
    ir_item=results_5.value;//赋值给变量
    String type="UNKNOWN";//任何类型都可赋值
    String typelist[18]={"UNUSED", "RC5", "RC6", "NEC", "SONY", "PANASONIC", "JVC", "SAMSUNG",
    "WHYNTER", "AIWA_RC_T501", "LG", "SANYO", "MITSUBISHI", "DISH", "SHARP", "DENON",
    "PRONTO", "LEGO_PF"};//红外接收遥控代码
    if(results_5.decode_type>=1&&results_5.decode_type<=17){
        type=typelist[results_5.decode_type];
        Serial.println("IR TYPE:"+type+" ");
        Serial.print(ir_item,HEX);
        if (ir_item == 0xffa857) {
            if (flag) {flag = false;
            } else {flag = true;
            }
            analogWrite(9,0);
        } else if (ir_item == 0xff30af) {
            if (!flag) {analogWrite(9,100);}
        } else if (ir_item == 0xff18e7) {
            if (!flag) {analogWrite(9,150);}
        } else if (ir_item == 0xff7a85) {
            if (!flag) {analogWrite(9,255);}
        }
        irrecv_5.resume();
    } else {
        if (flag) {digitalWrite(13,HIGH);
        if (round(checkdistance_3_2()) < 30) {
            analogWrite(9,(map(checkdistance_3_2(), 1, 30, 100, 255)));
        } else {
            analogWrite(9,0);
        }
    } else {
        digitalWrite(13,LOW);}}}}
```



项目简析



项目任务

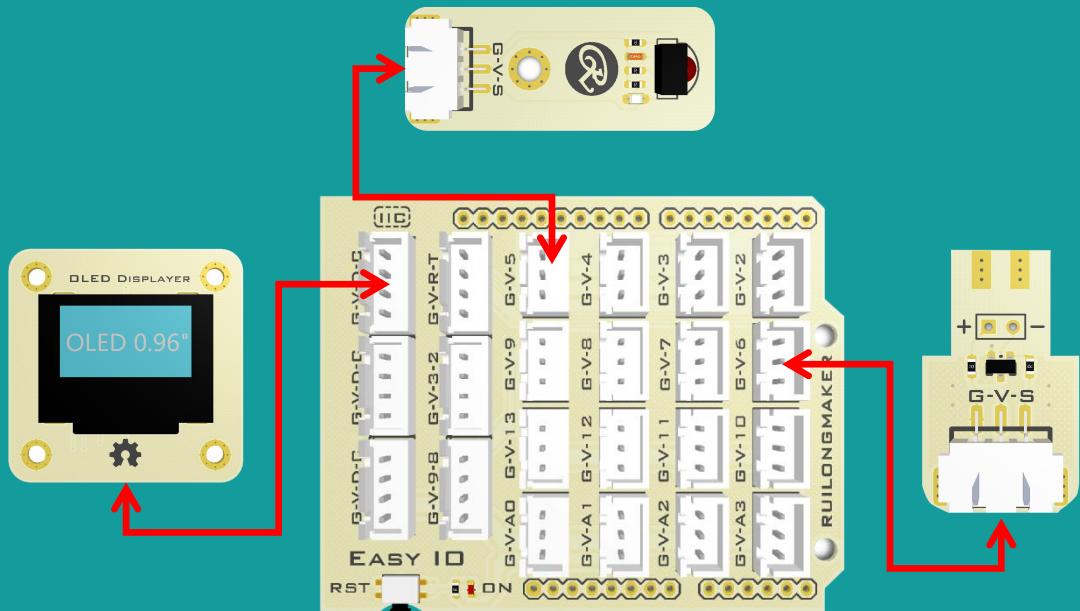
通过红外接收传感器、LED灯串模块，SSD1306显示屏，制作一个红外遥控调光灯，按下遥控器按键，改变LED灯串亮度，在SSD显示屏上以进度条的形式显示亮度级别。

思路分析

首先通过串口监视器记录下按键对应的编码，编写进度条函数，每个亮度对应一个进度条显示，依据编码的不同执行不同的亮度语句，调用不同的进度条显示函数。



硬件连线



序号	名称	连线	备注
1	红外接收传感器	D5	
2	LED灯串模块	D6	
3	SSD1306	GVDC	



程序编程

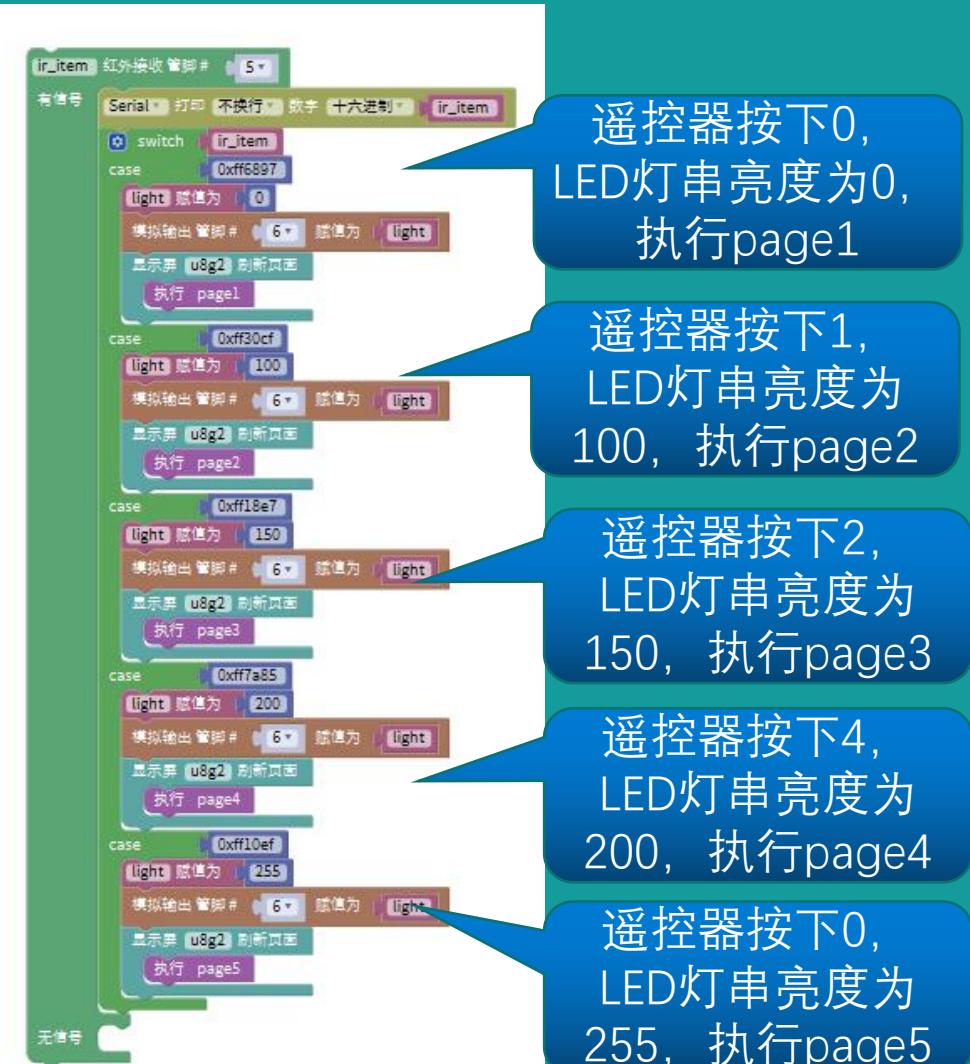
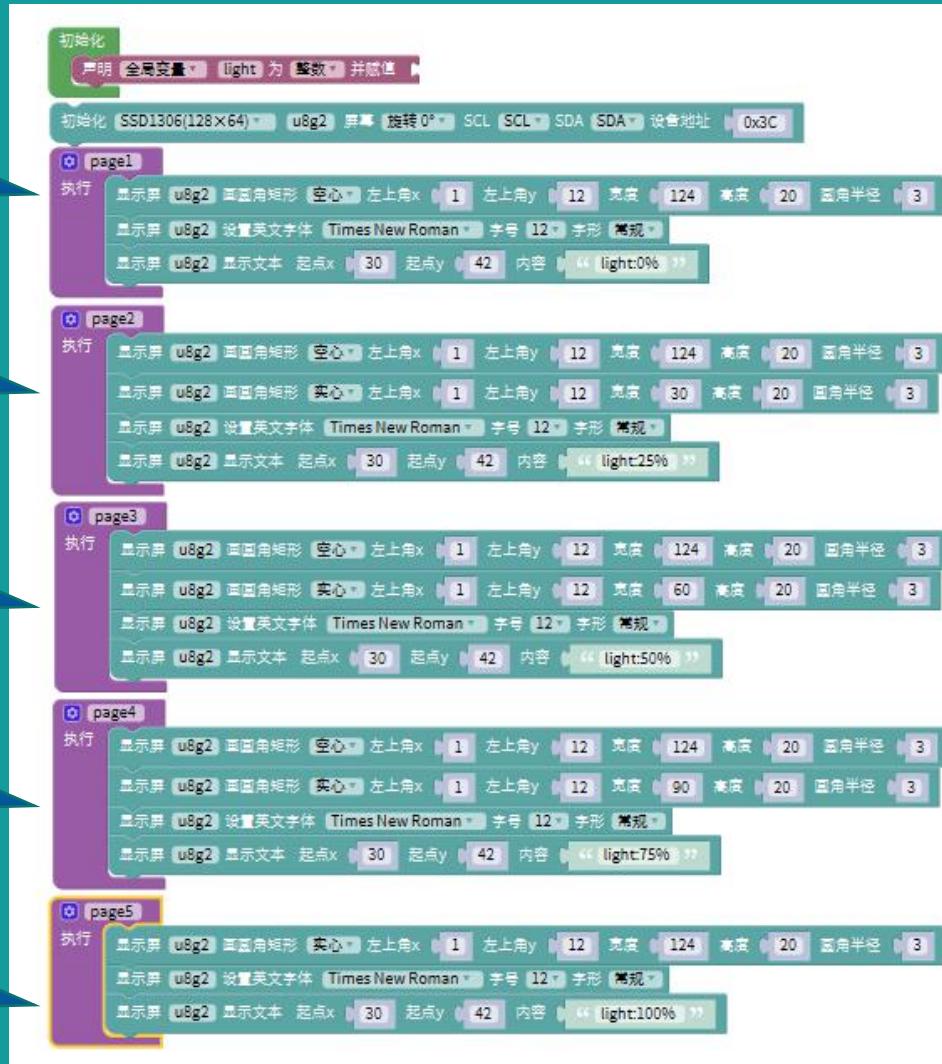
进度条显示
light: 0%

进度条显示
light: 25%

进度条显示
light: 50%

进度条显示
light: 75%

进度条显示
light: 100%



遥控器按下0,
LED灯串亮度为0,
执行page1

遥控器按下1,
LED灯串亮度为
100, 执行page2

遥控器按下2,
LED灯串亮度为
150, 执行page3

遥控器按下4,
LED灯串亮度为
200, 执行page4

遥控器按下0,
LED灯串亮度为
255, 执行page5



1、头文件引入

```
#include <IRremote.h>
#include <U8g2lib.h>
#include <Wire.h>
```

2、定义变量

```
volatile int light;
long ir_item;
```

3、创建对象

```
IRrecv irrecv_5(5);
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);
```

4、结构体变量

```
decode_results results_5;
```

5、显示页面函数

```
void page1() {
    u8g2.drawRFrame(1,12,124,20,3);
    u8g2.setFont(u8g2_font_timR12_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(30,42);
    u8g2.print("light:0%");}
void page2() {
    u8g2.drawRFrame(1,12,124,20,3);
    u8g2.drawRBox(1,12,30,20,3);
    u8g2.setFont(u8g2_font_timR12_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(30,42);
    u8g2.print("light:25%");}
void page3() {
    u8g2.drawRFrame(1,12,124,20,3);
    u8g2.drawRBox(1,12,60,20,3);
    u8g2.setFont(u8g2_font_timR12_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(30,42);
    u8g2.print("light:50%");}
    u8g2.print("light:100%");}
```

```
void page4() {
    u8g2.drawRFrame(1,12,124,20,3);
    u8g2.drawRBox(1,12,90,20,3);
    u8g2.setFont(u8g2_font_timR12_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(30,42);
    u8g2.print("light:75%");}
void page5() {
    u8g2.drawRBox(1,12,124,20,3);
    u8g2.setFont(u8g2_font_timR12_tf);
    u8g2.setFontPosTop();
    u8g2.setCursor(30,42);
    u8g2.print("light:100%");}
```

6、setup函数

```
void setup(){
    light = 0;
    Serial.begin(9600);
    irrecv_5.enableRIn();
    u8g2.setI2CAddress(0x3C*2);
    u8g2.begin();
    u8g2.enableUTF8Print();
}
```



1、loop函数

```
void loop(){
    if (irrecv_5.decode(&results_5)) {
        ir_item=results_5.value;
        String type="UNKNOWN";
        String typelist[18]={"UNUSED", "RC5", "RC6", "NEC", "SONY", "PANASONIC", "JVC", "SAMSUNG",
        "WHYNTER", "AIWA_RC_T501", "LG", "SANYO", "MITSUBISHI", "DISH", "SHARP", "DENON",
        "PRONTO", "LEGO_PF"};
        if(results_5.decode_type>=1&&results_5.decode_type<=17){
            type=typelist[results_5.decode_type];
        }
        Serial.println("IR TYPE:"+type+" ");
        Serial.print(ir_item,HEX);
        switch (ir_item) {
            case 0xff6897:
                light = 0;
                analogWrite(6,light);
                u8g2.firstPage();
                do
                {
                    page1();
                }while(u8g2.nextPage());
                break;
            case 0xff30cf:
                light = 100;
                analogWrite(6,light);
                u8g2.firstPage();
                do
                {
                    page2();
                }while(u8g2.nextPage());
                break;
        }
    }
}
```

```
case 0xff18e7:
    light = 150;
    analogWrite(6,light);
    u8g2.firstPage();
    do
    {
        page3();
    }while(u8g2.nextPage());
    break;
case 0xff7a85:
    light = 200;
    analogWrite(6,light);
    u8g2.firstPage();
    do
    {
        page4();
    }while(u8g2.nextPage());
    break;
case 0xff10ef:
    light = 255;
    analogWrite(6,light);
    u8g2.firstPage();
    do
    {
        page5();
    }while(u8g2.nextPage());
    break;
}
irrecv_5.resume();
} else {
}

}
```



项目简析



项目任务

通过红外接收传感器、红外遥控器、蜂鸣器模块制作一个简易的遥控钢琴。

思路分析

首先连接好红外接收传感器，上传红外接收管脚程序，打开串口监视器，按下红外遥控器按键，记录下按键对应的16进制编码，按键A,B,C,D用来控制节拍，按键左、中、右控制音高，按键1-7用来控制蜂鸣器发出声音。



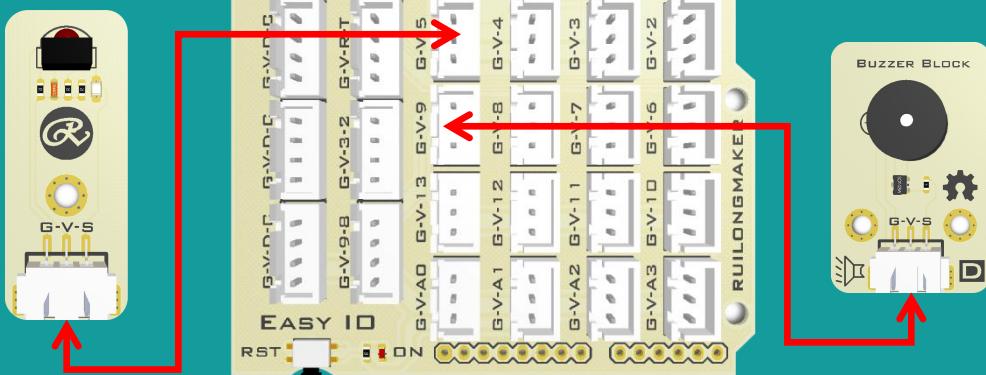
参考表格



按键	16进制编码	功能	按键	16进制编码	功能
A	FFA25D	半拍	1	FF30CF	DO
B	FF629D	一拍	2	FF18E7	RE
C	FFE21D	两拍	3	FF7A85	MI
D	FF22DD	四拍	4	FF10EF	FA
左	FFE01F	低音	5	FF38C7	SO
中	FFA857	中音	6	FF5AA5	LA
右	FF906F	高音	7	FF42BD	SI



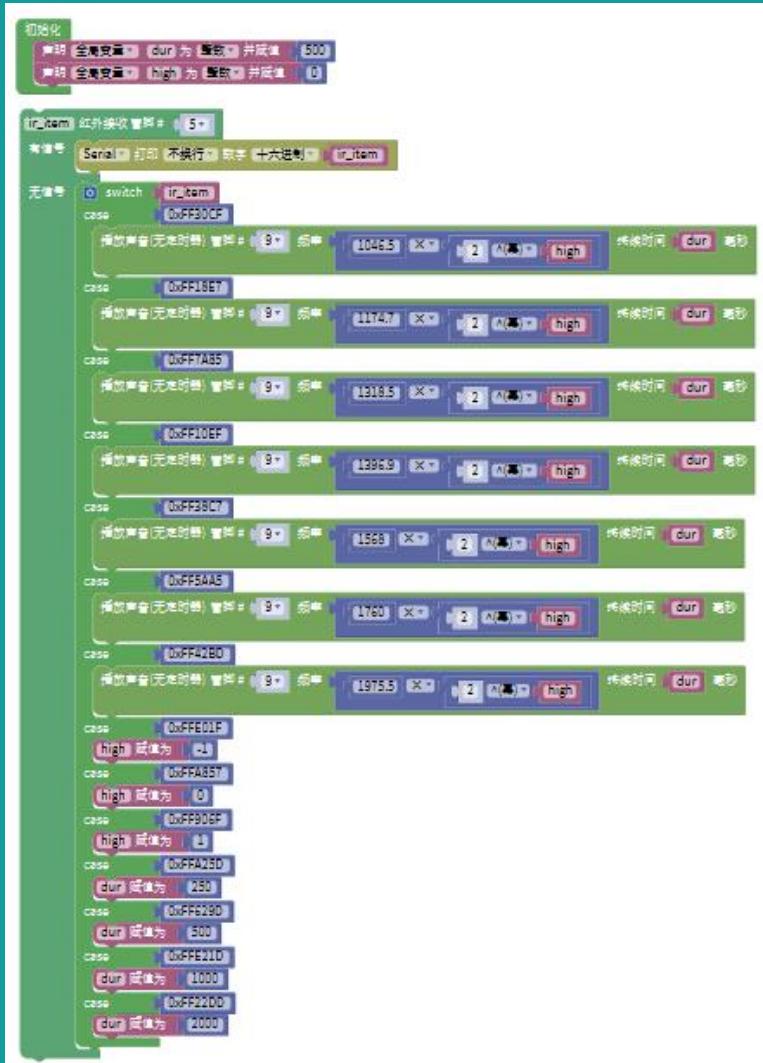
硬件连线



序号	名称	连线	备注
1	红外接收传感器	D5	
2	蜂鸣器模块	D9	



程序编程



- 1、初始化变量dur,high,分别用来控制拍数和音高。
- 2、假定500ms为一拍，半拍为250ms,两拍为1000ms,四拍为2000ms。
- 3、频率=基准频率 $\times 2^{\text{high}}$,当high=0时，频率为基准频率，当high=1时，频率为基准频率的两倍，声音高一个八度；当high=-1时，频率为基准频率的一般，声音低一个八度。
- 4、依据上表按下相应按键，可以实现简易的钢琴模拟功能。



1、头文件引入

```
#include <NewTone.h>
#include <IRremote.h>
```

2、定义变量

```
volatile int dur;
volatile int high;
long ir_item;
decode_results results_5;//结构体变量
```

3、创建对象

```
IRrecv irrecv_5(5);
```

4、setup函数

```
void setup(){
    dur = 500;
    high = 0;
    Serial.begin(9600);
    pinMode(9, OUTPUT);
    irrecv_5.enableIRIn();
}
```

5、loop函数

```
void loop(){
    if (irrecv_5.decode(&results_5)) {//尝试解码接收到的红外信号，如果有返回true,
        ir_item=results_5.value;//保存解码值
        String type="UNKNOWN";
        String typelist[18]={"UNUSED", "RC5", "RC6", "NEC", "SONY", "PANASONIC", "JVC", "SAMSUNG",
        "WHYNTER", "AIWA_RC_T501", "LG", "SANYO", "MITSUBISHI", "DISH", "SHARP", "DENON",
        "PRONTO", "LEGO_PF"};
        if(results_5.decode_type>=1&&results_5.decode_type<=17){
            type=typelist[results_5.decode_type];
        }
        Serial.println("IR TYPE:"+type+" ");
        Serial.print(ir_item,HEX);
        irrecv_5.resume();
    } else {

```

5、loop函数

```
switch (ir_item) {
    case 0xFF30CF:
        NewTone(9,1046.5 * pow(2, high),dur);
        break;
    case 0xFF18E7:NewTone(9,1174.7 * pow(2, high),dur);
        break;
    case 0xFF7A85:NewTone(9,1318.5 * pow(2, high),dur);
        break;
    case 0xFF10EF:NewTone(9,1396.9 * pow(2, high),dur);
        break;
    case 0xFF38C7: NewTone(9,1568 * pow(2, high),dur);
        break;
    case 0xFF5AA5:NewTone(9,1760 * pow(2, high),dur);
        break;
    case 0xFF42BD:NewTone(9,1975.5 * pow(2, high),dur);
        break;
    case 0FFE01F:high = -1;
        break;
    case 0FFA857:high = 0;
        break;
    case 0FF906F:high = 1;
        break;
    case 0FFA25D:dur = 250;
        break;
    case 0FF629D: dur = 500;
        break;
    case 0FFE21D:dur = 1000;
        break;
    case 0FF22DD:dur = 2000;
        break; }}}
```