基于 arduino 的跑酷游戏

沈玥 物理学系 17307110399

摘要：本实验利用 arduino 单片机和超声测距模块、无源蜂鸣器、按钮和 1602ALCD 液晶显示屏实现了一个跑酷游戏。跑酷游戏中包含障碍物、金币以及奖励标志。游戏会随着时间的推移而障碍物更为密集且移动速度更快。

一、课题设计

Arduino 是一款简便灵活的开源电子原型平台，提供了一种低成本的简单方法，以创建使用传感器与环境相互作用的设备执行器。它可以使用现有的电子元件或者其他控制器件，并通过代码实现交互，以此开发简单机器人、恒温器或者运动检测器。[1]

本实验利用 arduino 单片机和多种传感器，可以实现多种方式的数据输入，结合 LCD 显示屏实现对显示的实时操作的特点。基于此特性设计了基于 arduino 的跑酷游戏。

二、实验原理

1、超声测距模块 HC-HR04

超声测距模块有四个引脚，分别是 Vcc，Trig，Echo，GND。Trig 是距离测量出发引脚，只要 Trig 引脚至少有 $10\mu s$的高电平，超声波发送模块会自动发送 8 个 40KHZ 超声波脉冲，并自动检测是否有返回信号。如果有任何返回信号，Echo 引脚将输出高电平，高电平的持续时间是超声波从发射到返回的时间，此时，我们可以使用 pulseIn（）函数来获得距离测量的结果，并计算实际距离，整个过程如图 1 所示。

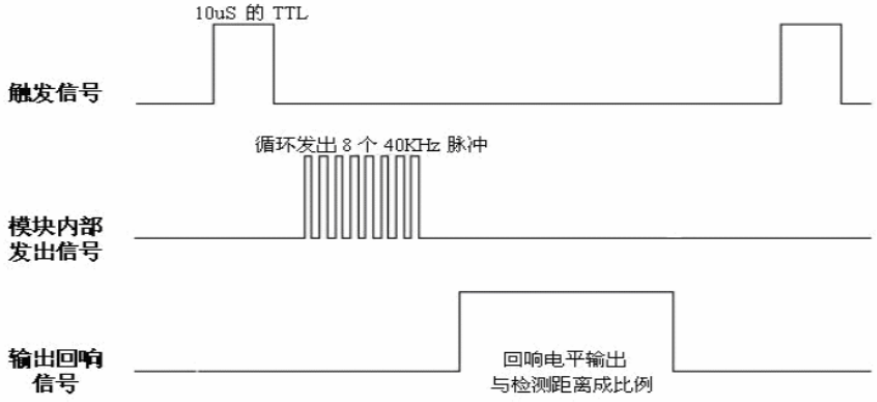

图 1　超声模块时序示意图

2、无源蜂鸣器

无源蜂鸣器内部没有振荡器，在通电时需要用方波驱动，不同频率的波形会发出相应的不同声音。无源蜂鸣器声音频率范围宽，可以应对跑酷游戏中的多种音效

3、LCD1602A

LCD1602A 是字符型液晶显示器，能够显示 32 个字符（2 行 16 列），每个字符对应一个 5×8 的矩阵。LCD1602A 本体有 16 个接口，接线较为复杂，但是利用 I2C 转接板可以大大简化接线。LCD 支持利用 CGRAM 地址存储 8 个自定义字符。利用 LCD1602A，可以完成跑酷游戏界面的显示。

三、实验过程

除单片机外，利用的元件有按钮，无源蜂鸣器，超声测距模块，LCD 显示屏。其中按钮用来控制游戏的开始，超声测距模块用作模拟滑动以切换小人的跑道的输入设备，无源蜂鸣器提供游戏过程中的音效。所有游戏的显示界面由 LCD 显示屏完成。

接线图如图 2 所示（此处 LCD 显示屏自带 I2C 接口，因此接线只有 4 条，并不代表真实接线位置）

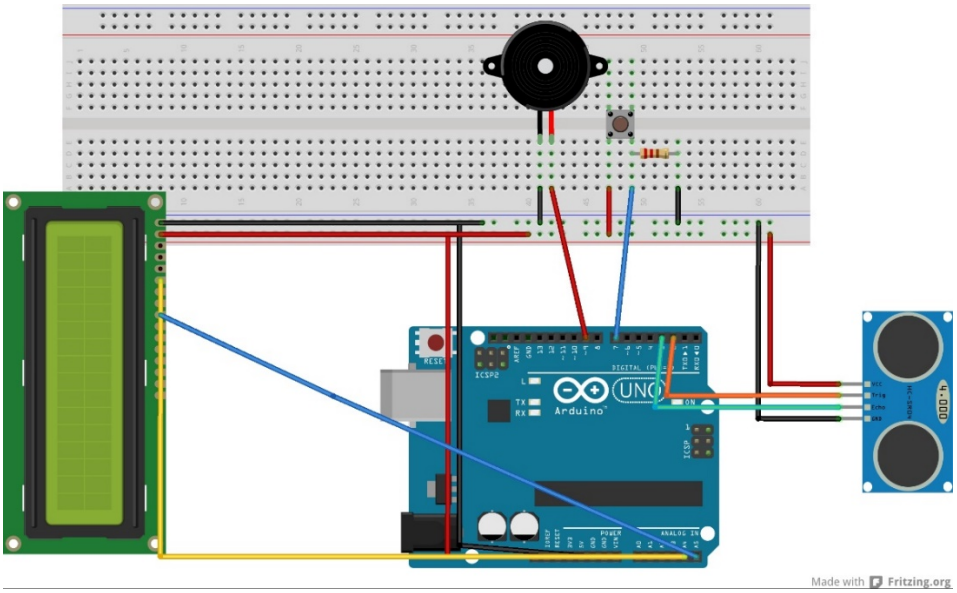

图 2　接线示意图

四、实验结果

游戏开始时，按下按钮，进入游戏界面，可以看到人物、障碍物和金币。

通过手的移动可以控制人物的上下位置躲避障碍物并吃到金币。

吃到奖励标志后，进入冲刺模式。

一段时间后，障碍物变得更为密集，难度加大。

五、实验分析

1、测试超声测距模块

在游戏中，玩家通过将手放在超声测距模块上，上下移动手的位置来实现对人物的控制。当手向下移动时，人物移动到 LCD 显示屏第二行，当手向上移动时，人物移动到 LCD 显示

屏第一行。因此，在程序中需要两个变量来记录前后两次距离的测量结果并进行比较，以此判断手是否发生了移动。但同时，人的手在正常悬浮时也会有些许位移，为了防止手的晃动引起的误操作，应设定前后两次距离之差大于某个值才可移动人物。因此需要测试具体阈值应设定为多少。

测试过程中，将手置于超声测距模块之上，隔一段时间后做上下移动的操作一次，利用串口监视器观察在非操作状态和操作时距离差的数量级。观察结果如图 3，上下滑动操作时的数据已经框出。可知正常状态下人手晃动导致的前后距离差不会超过 3cm，而试图操作时距离差达到 10cm 左右，因此大致设定阈值为 7.5cm。

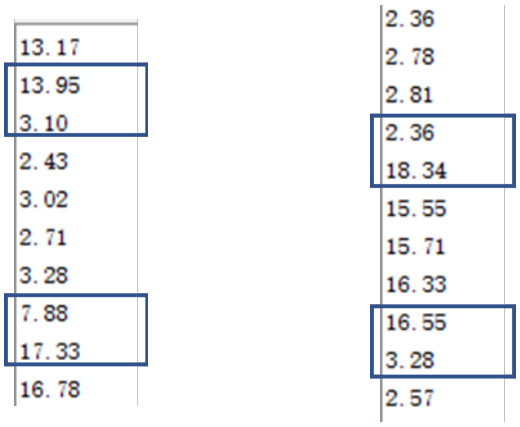

图 3　超声测距模块测试数据图

2、测试无源蜂鸣器模块

在游戏中，人物的移动，金币的获取，冲刺模式需要对应不同的音效，因此采用无源蜂鸣器，通过控制高低电平的持续时间实现不同的音效。

测试时分别用频率为 500Hz 和 50Hz 的两种方波驱动，发现当驱动频率更高时无源蜂鸣器发出声音的频率也更高。因此再获取金币时采用更高的频率，人物跳跃时采用较低的频率，冲刺模式时两种频率交替发出以达到不同的效果。

3、跑酷程序的实现

游戏中涉及了三种标识物——障碍，金币和奖励（冲刺模式）标志。用三个数组标记。对这三个数组有不同的处理。

障碍物是跑酷游戏的基础，玩家在游戏过程中不得触碰障碍物。障碍物包含两个属性:长度和位置。位置记录每一组障碍物出现的起始位置，长度记录这一组障碍物包含的数量，如图所示。这两个属性由随机数控制生成。由于 1602A 只包含两行，因此为了防止障碍物在随机生成的过程中出现两行轨道都被障碍物填充的死局，应通过控制随机数范围保证障碍

物长度小于障碍物位置间隔，并且使障碍物在上下行交替出现。利用数组记录这些数据，这样每组障碍物会拥有一个编号，两个属性，方便后续的循环读取。

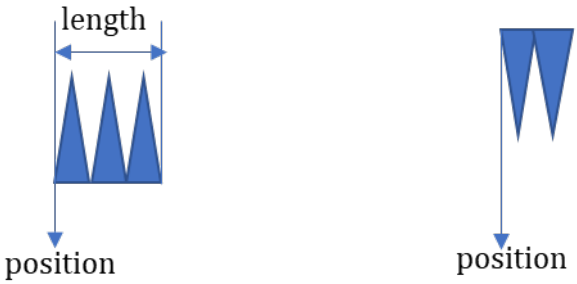

图 4　障碍物示意图

金币的原理和障碍物相似，区别在于人物碰撞后并不使游戏结束，而是加分。金币同样也有两个属性标记，由随机数控制生成。但是为了防止金币和障碍物冲突，在程序中使金币的位置处于同一行的两组障碍物（即编号相差为 2 的障碍物组）之间，并且其长度受到控制。这样产生的金币不会与障碍物冲突，且相比金币出现频率更低。

奖励标志用和金币一样的方式插在金币之间，在人物碰撞到奖励标志时，进入冲刺模式。此时令显示屏上显示 dash 字样。在 dash 模式过程中，玩家会自动跳过一定数量的障碍物组，并且得分大大增加。为了实现跳过障碍物的功能，假设正常循环下读取到第 k 组障碍物，那么冲刺模式结束后将读取第 k+5 个障碍物，以此跳过 5 组障碍物实现冲刺。

在主程序部分，利用变量 i 记录人物的移动的距离，i 值随着时间的推移自动增加，表现在 lcd 显示屏幕上，人物位于最左端保持不变，相当于是障碍物向人物靠近了距离 i。循环变量 k 读取障碍物组编号，第 k 组障碍物经过 i 的距离后会显示在屏幕上时，根据其位置和长度属性在显示屏上输出障碍物的样式。随着 i 的增加，障碍物会越来越接近人物直到被越过。以此形成跑酷的游戏。

总结主程序核心循环算法如下：

A）利用超声测距模块的数据，比对前一时刻的位置数据，判断是否需要让小人移动，若移动则发出移动音效

B）人物位置 i+1，得分+1

C）对障碍物、金币、奖励标志的编号 k+dashcounter*5 进行循环，如果有标志的位置-人物位置 i<16，则判断该标志应该出现在屏幕上，根据其长度属性连续输出这组标志。

D）根据人物位置 i 和标志的位置进行碰撞判断，分为如下三种情况

碰到障碍物：激活游戏结束变量 over；

碰到金币：发出碰到金币的音效，得分+2；

碰到奖励：dashcounter+1，得分+15，屏幕上显示 dash 字样并发出相应音效；

E）延迟一定时间后，进入下一次循环

4、游戏随时间的改变

随着时间推移，跑酷游戏的难度增加，体现在障碍物的间隔变短，障碍物变长，移动速度变快。因此在数组生成时还需要根据障碍物组编号控制两个属性的值的范围。在主程序部分的延迟时间上，则用 timecounter 进行控制。举例来说，对第 k 个障碍物，可以在随机生成其位置时令

$$position[k] = position[k-1] + random(\max\left(1{,}6 - \frac{k}{10}\right), \min\,(5{,}10 - \frac{k}{10}))$$

$$length[k] = random(1, \max\left(5{,}3 + \frac{k}{10}\right))$$

这样随着 k 的增加，障碍物组间的间隔会从 5-10 逐渐降至 1-5，而长度则有 1-3 增加至 1-5。延迟时间也是用同样的方法控制，以此随着时间的推移，障碍物的移动速度会越来越快直至阈值。

5、界面设计

利用 1602ALCD 液晶显示屏的自定义字符功能可以对界面实现美化。如原理部分所示可以用一个 5×8 的矩阵记录输出的字符，1 代表显示 0 代表不显示。以此设计了如下的人物、障碍物、金币、奖励标志图样。

图 5　自定义字符示意图

具体实验效果请参看附件视频。

六、参考文献

[1]中文维基百科，arduino

附：IDE 全部代码

```
#include <Wire.h>
#include "LiquidCrystal_I2C.h"
const int TrigPin = 2;
const int EchoPin = 3;
int val=1;
float distancei=0.0;
float distancef=0.0;
LiquidCrystal_I2C lcd(0x27,16,2);
int buzzer=9;
int beginsw=7;
int ob[100][2]={0};
int co[100][2]={0};
int bonus[100][2]={0};
int i,j,k,t,l=0;
int N=50;
int over=0;
int phase=0;
int judge=1;
int score=0;
int timecounter=0;
int dashcounter=0;

byte peakdown[8]={
  B00100,
  B01010,
  B01010,
  B01010,
  B10001,
  B10001,
  B10001,
};
byte peakup[8]={
  B10001,
  B10001,
  B10001,
  B10001,
  B01010,
  B01010,
  B01010,
  B00100,
};
byte coin[8]={
  B00000,
  B00000,
  B11111,
  B10001,
  B10001,
  B11111,
  B00000,
  B00000,
};
byte hero[8]={
  B00000,
  B01110,
  B01110,
  B00100,
  B11111,
  B00100,
```

```
    B01010,
    B10001,
};
byte star[8]={
    B00000,
    B10101,
    B01110,
    B11111,
    B01110,
    B10101,
    B00000,
    B00000,
};

void frequence_1(void){
    int i;
    for(i=0;i<10;i++){
        digitalWrite(buzzer,HIGH);
        delay(10);
        digitalWrite(buzzer,LOW);
        delay(10);
    }
}
void frequence_2(void){
    int i;
    for(i=0;i<200;i++){
        digitalWrite(buzzer,HIGH);
        delay(0.2);
        digitalWrite(buzzer,LOW);
        delay(0.2);
    }
}
void frequence_3(void){
    int i;
    lcd.clear();
    for(i=0;i<30;i++){
        frequence_1();
        frequence_2();
        lcd.clear();
        lcd.setCursor(4,1);
        lcd.print("DASH!!!!");
    }
}

void setup()
{
    lcd.init();
    lcd.backlight();
    lcd.createChar(0,peakup);
    lcd.createChar(1,peakdown);
    lcd.createChar(2,coin);
    lcd.createChar(3,hero);
    lcd.createChar(4,star);
    Serial.begin(9600);
    pinMode(TrigPin, OUTPUT);
    pinMode(EchoPin, INPUT);
    pinMode(buzzer,OUTPUT);
    pinMode(beginsw,INPUT);

}
```

```
void loop(){
  int bg=digitalRead(beginsw);
  if (bg){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Game start!");
    ob[0][0]=5;ob[0][1]=6;
    co[0][0]=4;co[0][1]=5;
    for(i=1;i<100;i++){
      ob[i][0]=ob[i-1][1]+random(max(1,6-phase/5),max(10-phase/5,5));
      ob[i][1]=ob[i][0]+random(1,min(3+phase/10,5));
      co[i][0]=ob[i][0]+ob[i-1][1]+1;
      co[i][1]=co[i][0]+random(1,4);
      bonus[i][0]=co[i][0]+co[i-1][1]+random(2,5);
      bonus[i][1]=bonus[i][0]+1;
      phase=phase+1;
    }
    delay(1000);
    lcd.clear();
    i=0;k=0;over=0;phase=0;score=0;timecounter=0;dashcounter=0;
    while(1){
      lcd.clear();
      lcd.setCursor(0,val);
      lcd.write(3);
      distancei=distancef;
      digitalWrite(TrigPin, LOW);
      delayMicroseconds(2);
      digitalWrite(TrigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(TrigPin, LOW);
      distancef = pulseIn(EchoPin, HIGH) / 58.00;
      Serial.println(distancef);
      for(k=0;k<N;k++){
        if(ob[k+dashcounter*5][0]-i<16){
          lcd.setCursor(max(0,ob[k+dashcounter*5][0]-i),(k+dashcounter*5)%2);
          for(l=ob[k+dashcounter*5][0]-i;l<ob[k+dashcounter*5][1]-i;l++){
            if (l>=16) break;
            else if (l>=0) {
              if((k+dashcounter*5)%2==0) lcd.write(0);
              else if ((k+dashcounter*5)%2==1) lcd.write(1);
```

```
                    }
                  }
                  if(co[k+dashcounter*5][0]-
i<16){

lcd.setCursor(max(0,co[k+dashcounter*5][
0]-i),(k+dashcounter*5)%2);

for(l=co[k+dashcounter*5][0]-
i;l<co[k+dashcounter*5][1]-i;l++){
                        if (l>=16) break;
                        else if (l>=0) {
                          lcd.write(2);
                      }
                    }
                  }

if(bonus[k+dashcounter*5][0]-i<16){

lcd.setCursor(max(0,bonus[k+dashcounter
*5][0]-i),(k+dashcounter*5)%2);

for(l=bonus[k+dashcounter*5][0]-
i;l<bonus[k+dashcounter*5][1]-i;l++){
                        if (l>=16) break;
                        else if (l>=0) {
                          lcd.write(4);
                      }
                    }
                  }

if(ob[k+dashcounter*5][0]<=i&&i<ob[k+da
shcounter*5][1]&&(k+dashcounter*5)%2==val){
                        over=1;
                        lcd.setCursor(0,val);
                        lcd.print("!");
                    }

if(co[k+dashcounter*5][0]<=i&&i<co[k+da
shcounter*5][1]&&(k+dashcounter*5)%2==val){
                        frequence_2();
                        lcd.setCursor(0,val);
                        lcd.write(3);
                        score=score+2;
                    }

if(bonus[k+dashcounter*5][0]<=i&&i<bonu
s[k+dashcounter*5][1]&&(k+dashcounter*
5)%2==val){
                        frequence_3();
                        lcd.clear();
                        dashcounter++;
                        score=score+15;
                    }
                }

            }
            if(over){
```

```
            delay(1500);                                    score++;
            lcd.clear();                                    i=i+1;
            lcd.print("Game over!");                       timecounter++;
            lcd.setCursor(0,1);
            lcd.print("Score:");                        }
            lcd.print(score-ob[1][0]);                }
            break;
        }
        if                  ((distancef-          }
distancei)>4&&val==1){
            lcd.setCursor(0,val);
            lcd.print(" ");
            val=0;
            lcd.setCursor(0,val);
            lcd.write(3);
            frequence_1();
        }
        else                if((distancei-
distancef)>4&&val==0){
            lcd.setCursor(0,val);
            lcd.print(" ");
            val=1;
            lcd.setCursor(0,val);
            lcd.write(3);
            frequence_1();
        }


        delay(min(300,400-
timecounter/10));
```