蓝桥杯树莓派模拟题解析一:智能灯

文/山东省招远第一中学新校微机组 牟晓东

蓝桥杯大赛是工信部人才交流中心举办的全国性专业信息技术赛事，为了更加全面和科学地评测学生的科技素养、逻辑思维和编程能力，从2021年的第12届起，青少组地区选拔赛开始采用STEMA评测考试。在11～18岁青少年创意编程大赛的STEMA测评Raspberry Pi树莓派-高级组中，编程题往往会被精心设置一定难度的“坑儿”，需要参赛者仔细审题并慎重作答，《电脑报》将通过解析一系列模拟题让大家更好地理解大赛的新变化。

第一题“智能灯”(难度系数1,18个计分点):

随着科技的发展，越来越多的设备都拥有了自己的“智能”，现在我们就来制作一个可以自己调节亮度的“智能灯”。

硬件准备:

1个光敏传感器【接入A0#管脚】,1个LED小灯【接入5#管脚】,1个模数转换模块【接入专用接口】,杜邦线若干(公对母,母对母,公对公)

编程实现:

(1)程序开始时,LED灯点亮;

(2)当环境光线变暗时,LED灯的亮度自动调高变亮;

(3)当环境光线变亮时,LED灯的亮度自动调低变暗;

(4)如此循环。

判断标准:

3分:实现“编程实现”中的(1);　6分:实现“编程实现”中的(2);

6分:实现“编程实现”中的(3);　3分:完全符合题意。

1.审清题意,抓住关键点

“智能灯”的关键考点是“自己调节亮度”,通过光敏传感器来监测周围环境的光线值范围，作为控制信号对LED灯进行亮度调节。

(1)光敏传感器有两种信号输出:D(Digital)数字和A(Analog)模拟,本题需要使用的是模拟信号,数字信号的两个值(0和1)无法实现信号的梯度大范围变化,所以要安装模数转换模块。

(2)编程时对LED灯进行“点亮”和“熄灭”有多种方法(包括“智能硬件”→“常用”中的“控制2号小灯亮”和“基础”中的“设置GPIO2为有电”),但要实现亮度的明暗程度调节只能使用“基础”中的“控制18号GPIO输出PWM1000”语句来实现。PWM即“Pulse Width Modulation”,意思是“脉冲宽度调制”,在开源硬件编程中通常用来控制电机的转速或LED灯的明暗程度(类似于武侠小说中“使用三成功力”的描述)。

(3)LED灯的PWM值范围是0-3000，而光敏传感器所检测到的周围环境光线模拟值是0-32767,二者间必须要使用“映射”进行数据比例转换后才能去控制LED的亮度控制。

2.实验器材的安装

按照题目要求,首先将一支LED灯插入扩展板的5#管脚,特别要注意“长腿为正极、短腿为负极”;然后将模数转换模块的长针脚插入IIC区域，注意标注一定要正确对应；接着将光敏传感器插入24号插孔,同样要特别注意四个针脚的标注,VCC代表电源正极,GND代表接地，模拟信号端连接的是扩展板A0#管脚;最后,给树莓派接通电源,启动操作系统(如图1)。

1

3.编程实现“智能灯”

(1)进入平台编程环境

浏览器访问http://www.gdwrobot.cn/登录古德微机器人平台,点击“设备控制”切换至“积木”编程区。

(2)光敏传感器模拟信号的处理

建立变量“光线强度”,为其赋值为“基础”中的“从ADS 0 获取模拟信号”，因为光敏传感器模拟信号端是插接在扩展板24号的A0#管脚;同时,可构建使用“输出调试信息‘光线强度’”语句,将该数据输出显示在LOG区。

(3)映射LED的PWM值

建立第二个变量“小灯PWM”,使用“映射数字”语句将“光线强度”的0-32767映射为自己的取值范围:0-3000。由于中间很多数据不能整除会产生小数,因此需要在前面添加一个“获取整数”模块;同样是为了程序调试的方便,再构建使用“输出调试信息‘小灯PWM’”语句。

(4)控制LED灯的亮度

添加“控制5号GPIO输出PWM”语句,其参数设置为变量“小灯PWM”,实现从“光线强度”到“小灯PWM”控制LED灯亮度的功能。

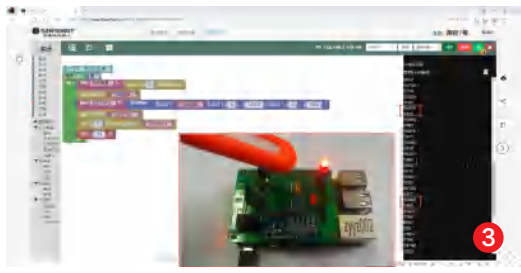
(5)其他细节

将以上语句全部放置于“重复当真”的循环结构中,同时别忘记在循环体最后添加一条“等待0.1秒”语句，目的是防止该循环过快，大量占用系统资源。另外,在循环结构开始之前要添加“控制5号小灯亮”语句,实现题目第(1)问(程序开始时点亮LED灯)的要求(如图2)。


2

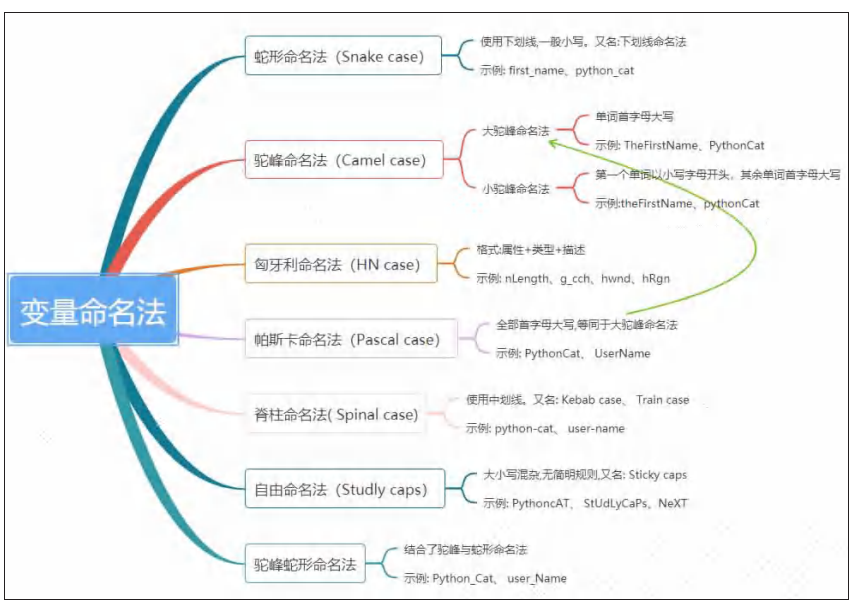
4.运行程序,检测“智能灯”

点击“连接设备”,出现五个绿色对钩提示,说明与树莓派操作系统连接成功;接着点击“运行”按钮,可以观察到LED灯被点亮。此时,借助小挡板将光敏电阻进行部分遮盖(模拟环境光线变暗),LED灯的亮度开始提高，同时在LOG区也分别显示有“光线强度”值(如15476、15998等)和“小灯PWM”值(如1416、1464等)(如图3)。如果再尝试使用手机的“手电筒”去照射光敏传感器的话,也会观察到LED灯的亮度会降低——需要将LED灯与光敏传感器用挡板“隔离”(手电筒的光线会影响到对LED亮度的观察)。


3

最后将程序按照要求保存为“01.txt”,下载并上传至考试系统中即可。

小常识——变量的命名法



各命名法思维导图

在编程中我们都会遇到自定义变量名的情况,随着程序的复杂就需要复杂的变量名表达更丰富的含义,这就会需要用到多个单词或符号。英语习惯使用空格来间隔开单词，然而空格一般在编程语言有特殊的意义，用在变量名中会带来一些麻烦,所以程序员们就创造出了各种命名法。总体而言,这些命名法都要克服单词间的空格，从而把不同单词串联起来，最终达到创造出一种新的“单词”的效果。

常见的命名法有:蛇形命名法(Snake case)、驼峰命名法(Camel case)、匈牙利命名法(HN case)、帕斯卡命名法(Pascal case)、脊柱命名法(Spinal case)、自由命名法(Studly caps)。

如果按照受众量与知名程度排名，毫无疑问排前两位的是驼峰命名法和蛇形命名法。在Python中一般对变量名推荐用蛇形命名法,毕竟Python自己就是一条蟒蛇嘛。而在类名、Type 变量、异常exception名这些情况下推荐用驼峰命名法。

驼峰命名法又分为小驼峰命名法(第一个单词以小写字母开头，其余单词首字母大写，如:theFirstName)和大驼峰命名法(所有单词首字母均大写,如:TheFirstName)。这样的变量名看上去就像驼峰一样在每个单词处起伏。

蛇形法是全由小写字母和下划线组成,在两个单词之间用下划线连接即可，如:first_name、last_name。这样变量名就由下划线像蛇一样连接起了所有单词。

这两种命名法有着各自的优缺点,用哪种更多是看程序员的习惯。

可读性:蛇形命名法用下划线拉大词距,更清楚易读;驼峰命名法的变量名紧凑,节省行宽。

易写性:驼峰命名法以大小写为区分,不引入额外的标识符;蛇形命名法统一小写,输入相对方便。

明义性:对于某些缩写成的专有名词,例如HTTP、RGB、DNS等等,一般习惯全用大写表示,但是如果严格遵循这两种命名法的话,须得只留首字母大写或者全小写,这样对原意都会造成一些“破坏”,有时候甚至让人感觉到别扭。如果保留全大写,IDE可能识别不准，反而会出现波浪提示。