

第一单元 点亮创客之路

流光沙漏

第3课

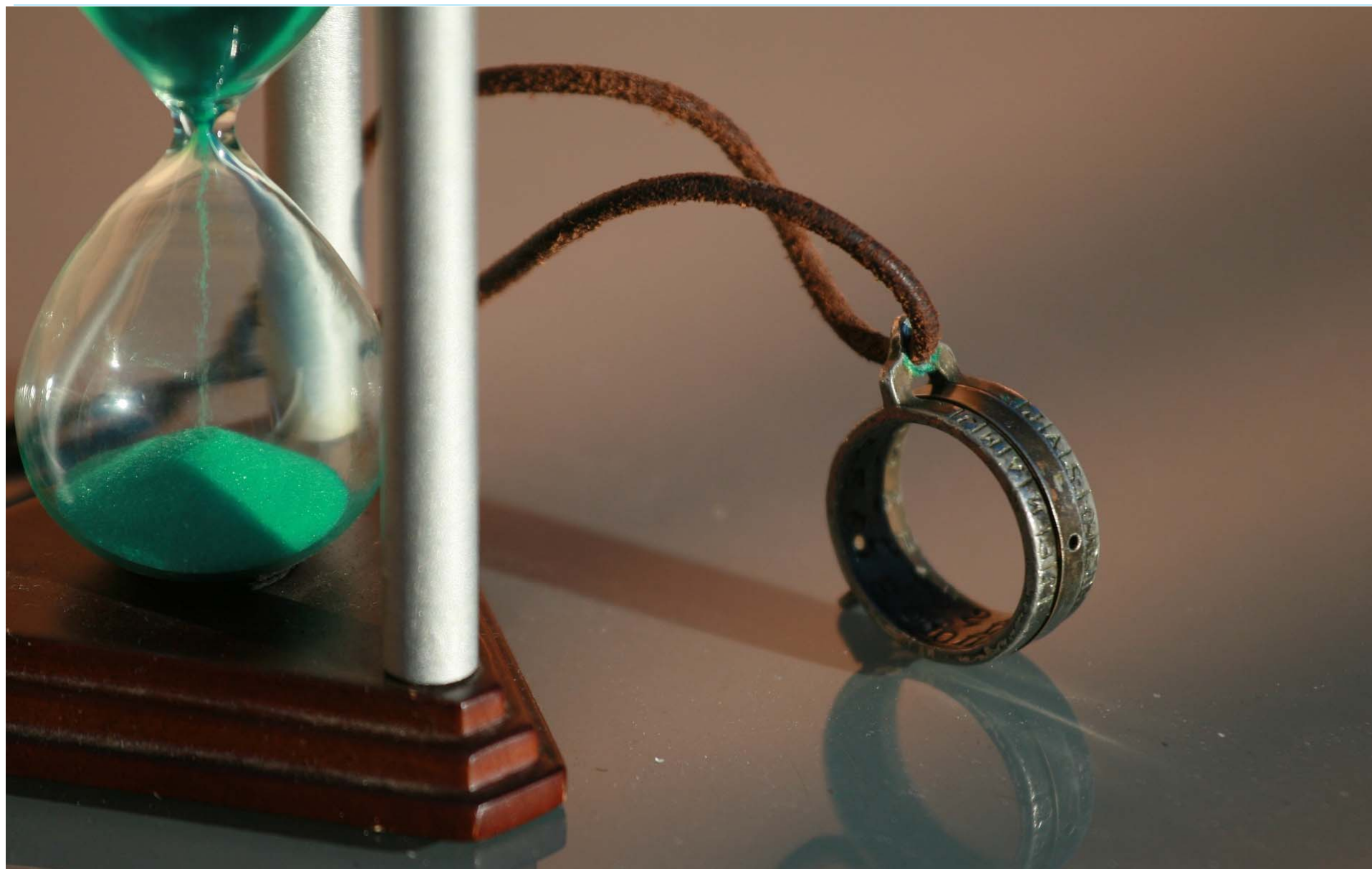


《创意电子入门》配套教学课件

3

流光沙漏

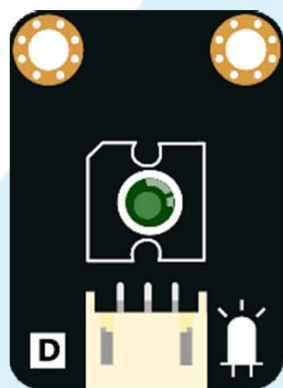
情境引入



3

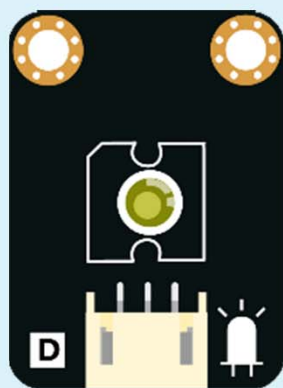
流光沙漏

模块清单



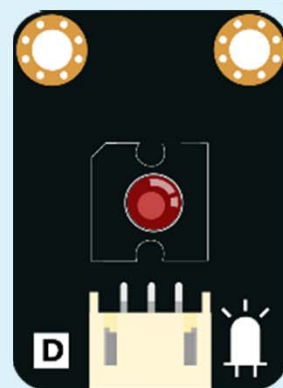
绿色 LED 灯

x1



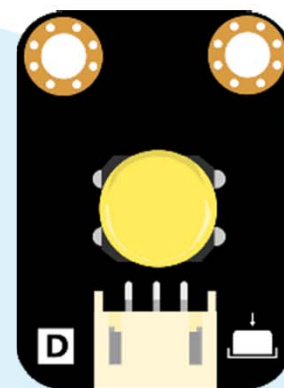
黄色 LED 灯

x1



红色 LED 灯

x1



黄色按钮x1



3

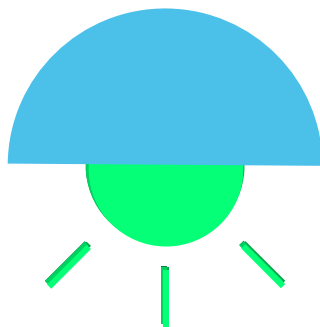
流光沙漏

简单任务 呼吸灯



任务发布

呼吸灯的灯光在微电脑的控制下，可以完成由暗到亮再由亮到暗的逐渐变化的过程，感觉像是在呼吸。请同学们使用LED灯，编写程序完成如下的实验效果：程序上传后，LED灯慢慢变亮。



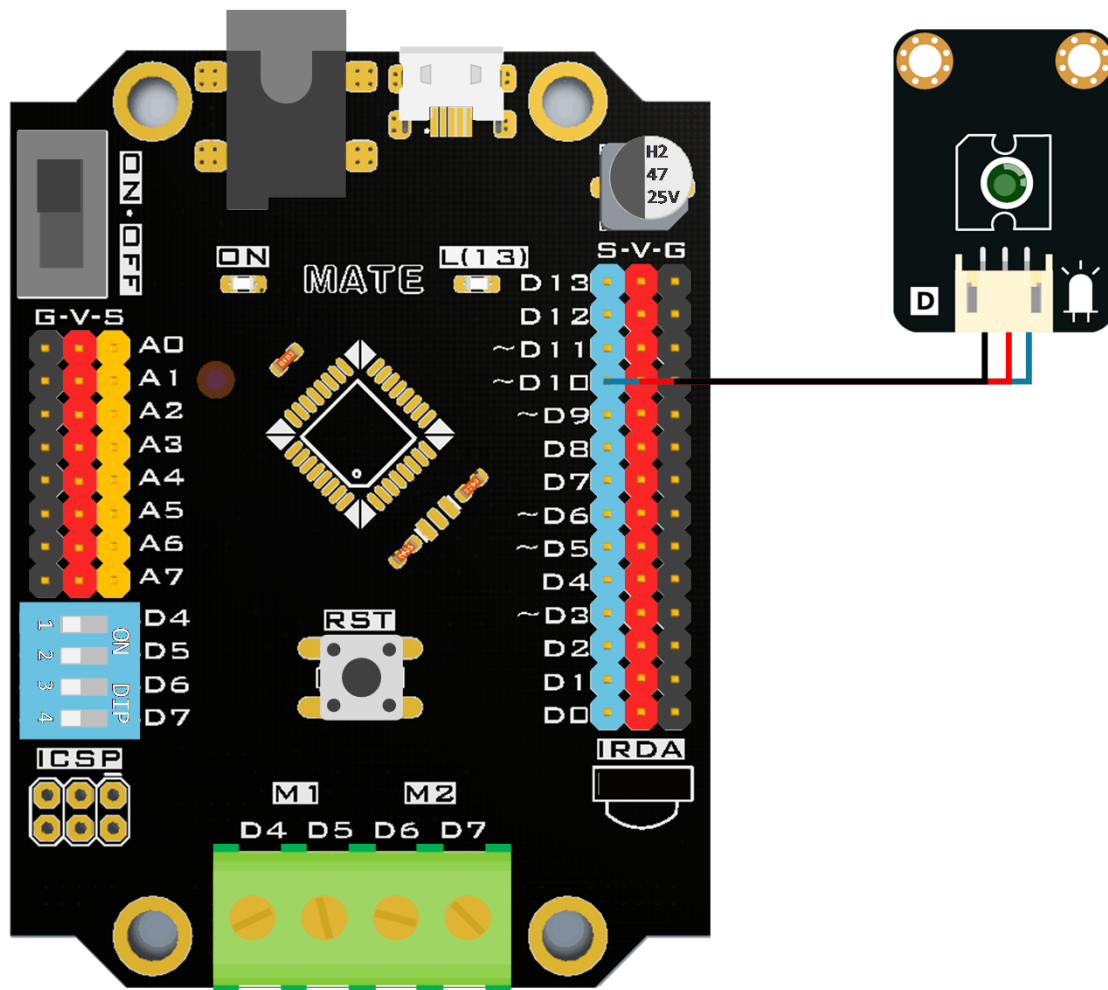
3

流光沙漏

简单任务 呼吸灯

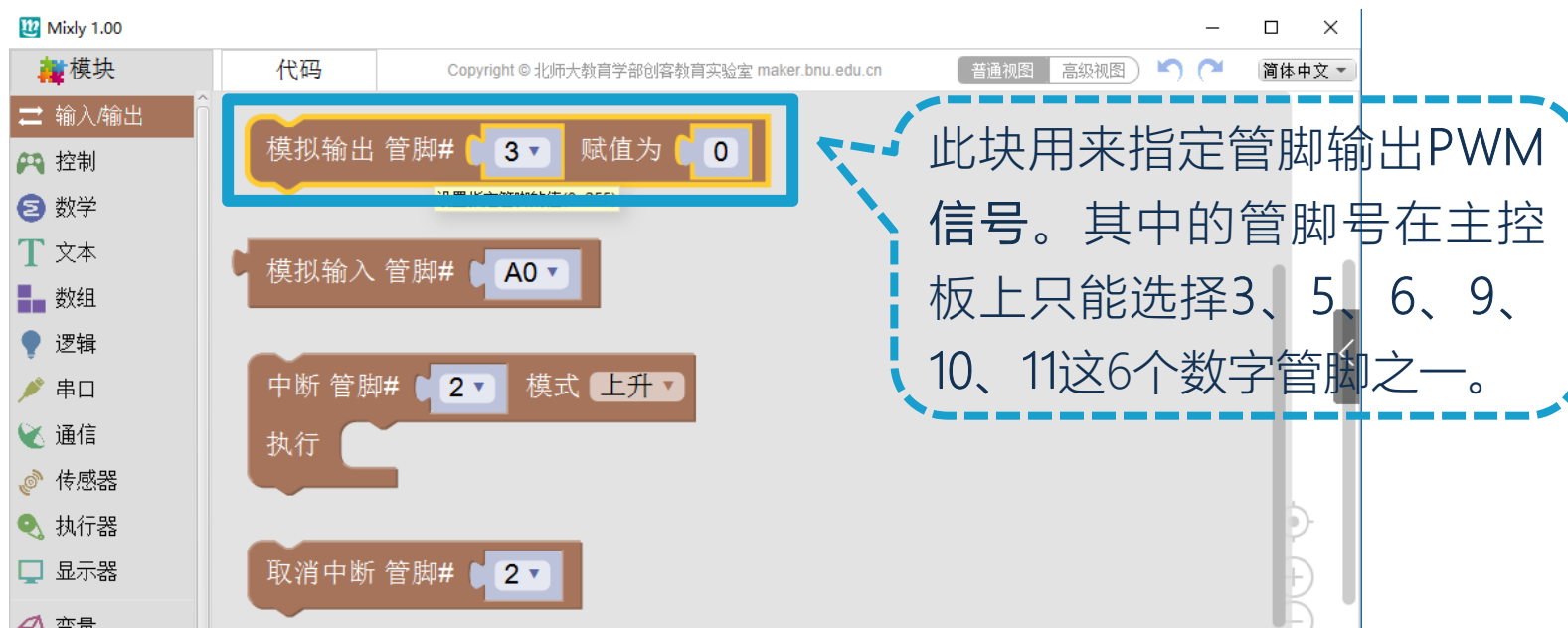


硬件连接





认识新代码块



- 模拟数值介于0~255之间，对应输出0~5V之间的仿真模拟电压值。



认识新代码块



- “使用i从1到10步长为1”代码块位于“控制”模块分类中。
- 其作用是：为变量(如i)从起始数(如1)到结尾数(如10)按指定的步长(间隔，如1)赋值，并执行指定的模块。

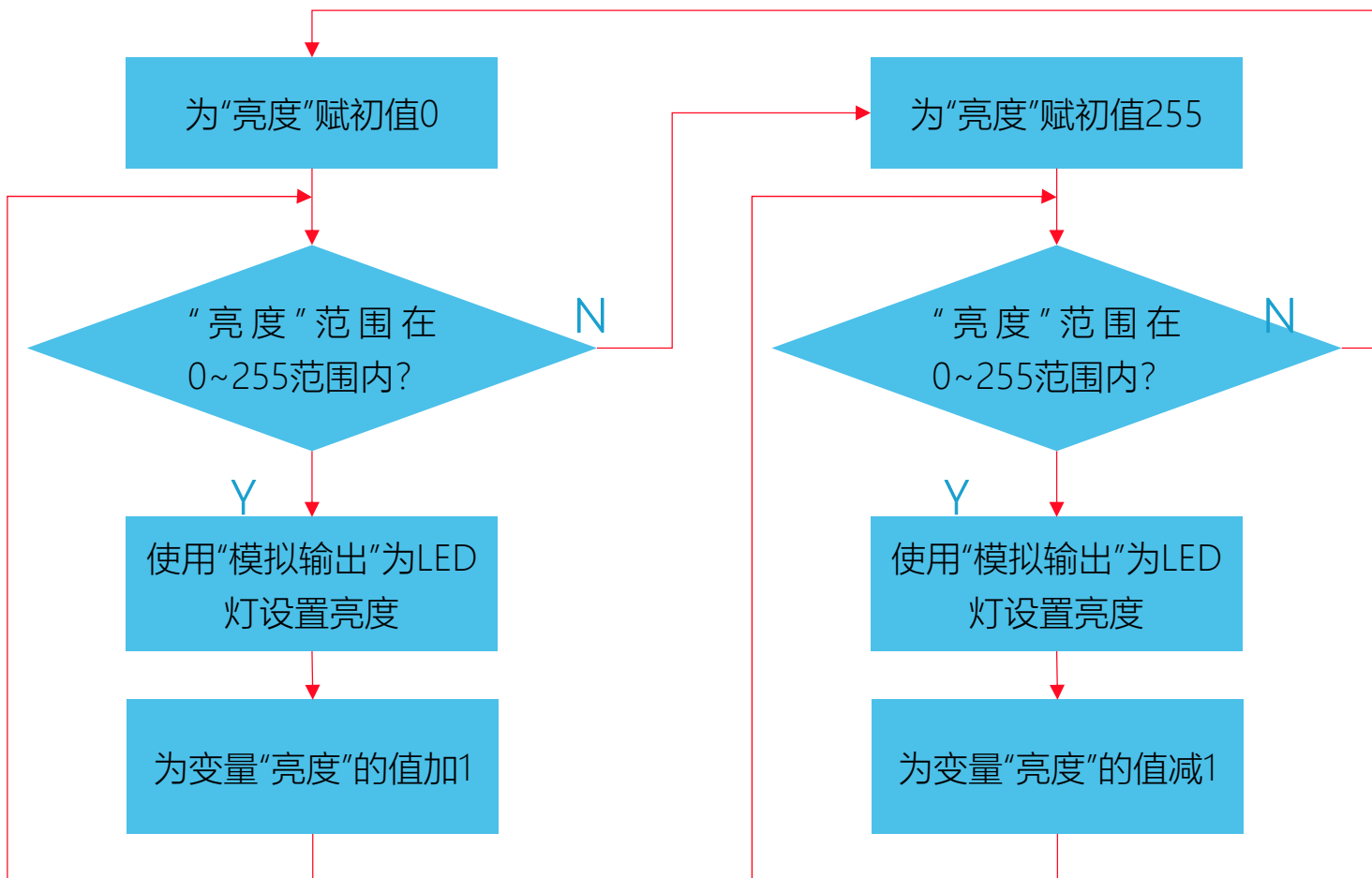
3

流光沙漏

简单任务 呼吸灯



编程思路



3

流光沙漏

简单任务 呼吸灯



软件编写

```
使用 亮度 从 0 到 255 步长为 1
执行
  模拟输出 管脚# 10 赋值为 亮度
  延时 毫秒 5

使用 亮度 从 255 到 0 步长为 -1
执行
  模拟输出 管脚# 10 赋值为 亮度
  延时 毫秒 5
```



- 很多情况下，程序需要重复执行一个程序段。这种结构被称为循环结构，这个被重复执行的程序段被称为循环体。



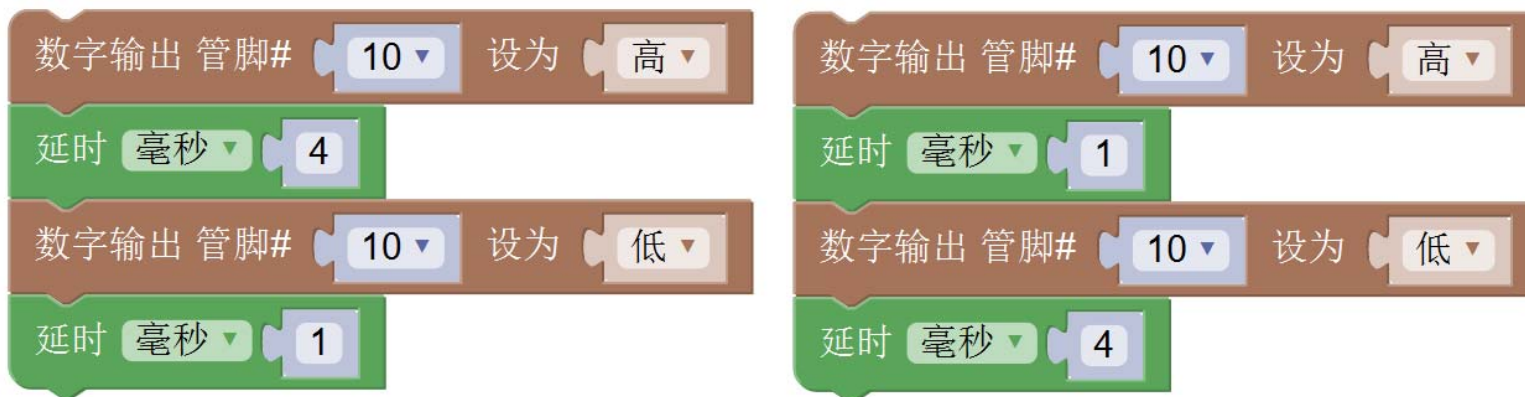
3

流光沙漏

知识点讲解 模拟输出



- 请尝试分别将下列代码上传到主控板上，观察效果：



- 左侧LED灯亮度 > 右侧LED灯亮度

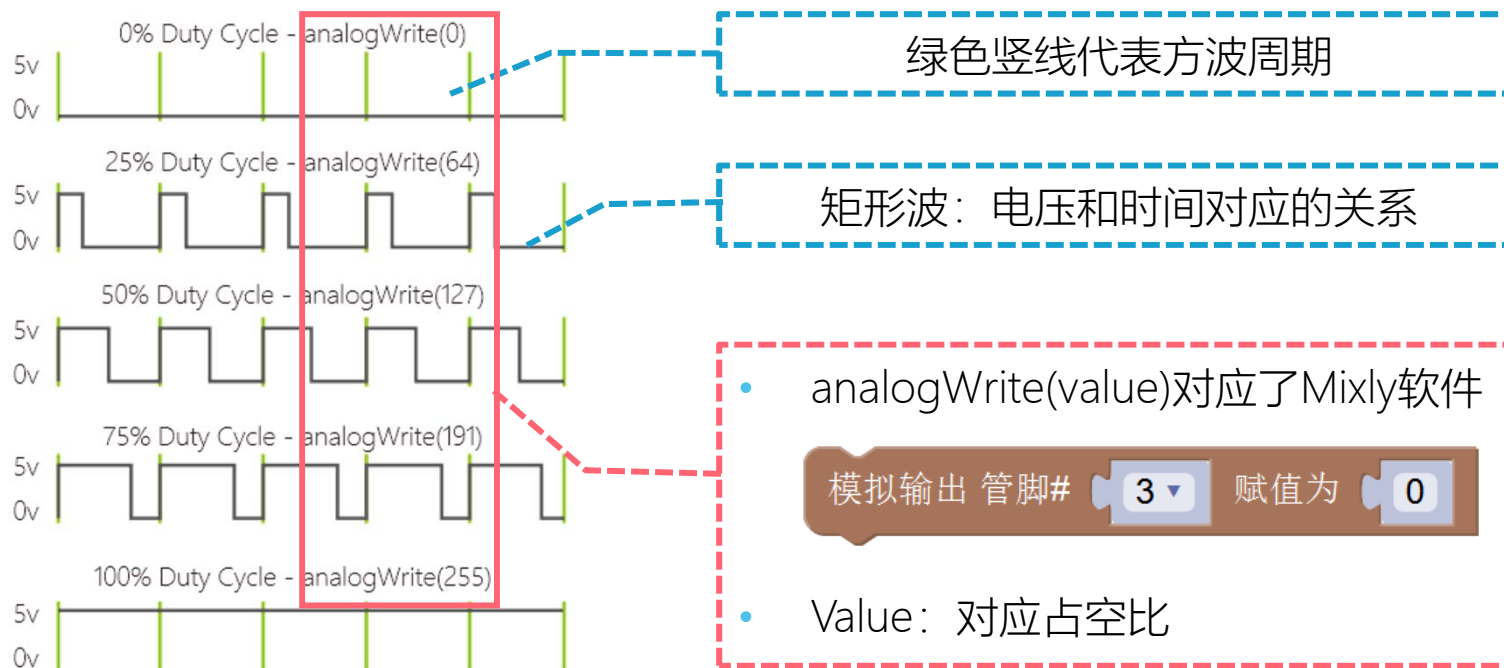
3

流光沙漏

知识点讲解 模拟输出



- Arduino可以通过**脉宽调制技术(PWM)**来控制LED灯的明亮度。
- 在Mate主控板上，有六个数字管脚标有“~”，这些管脚都具有PWM功能。



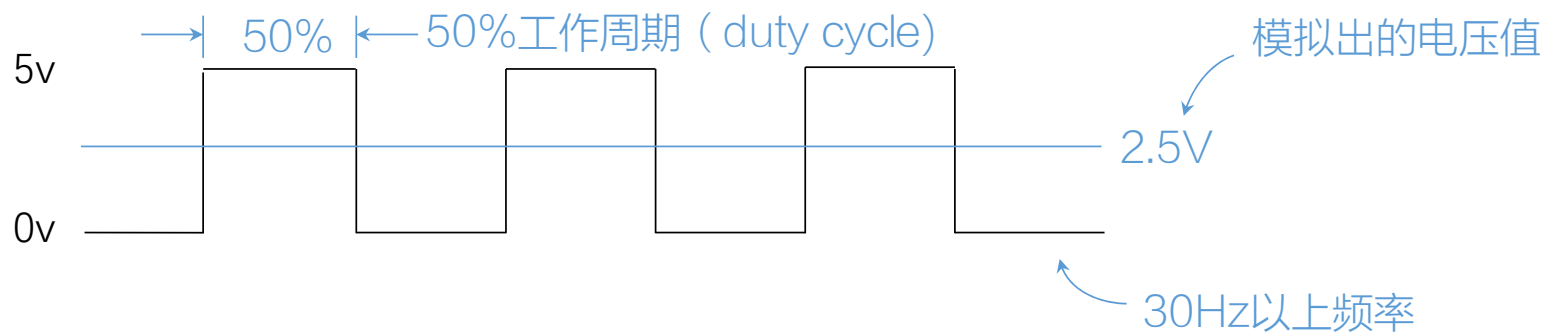
3

流光沙漏

知识点讲解 模拟输出



- 下图示例中占空比为50%:



3

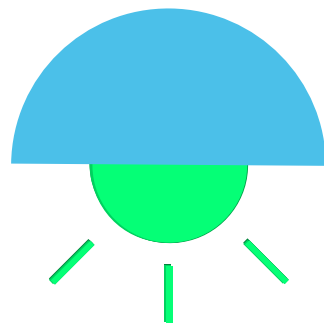
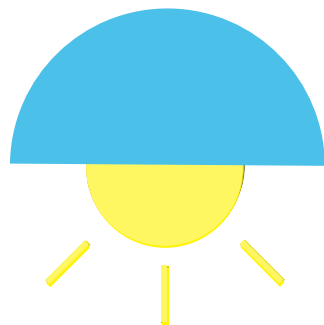
流光沙漏

可扩展任务 流光沙漏



任务发布

流光沙漏的两个LED灯的灯光互补，即当一个逐渐变暗时，另一个逐渐变亮。请同学们使用两个LED灯，编写程序，用LED灯的亮度模拟沙漏中的沙子。LED灯越亮表示沙子越多。



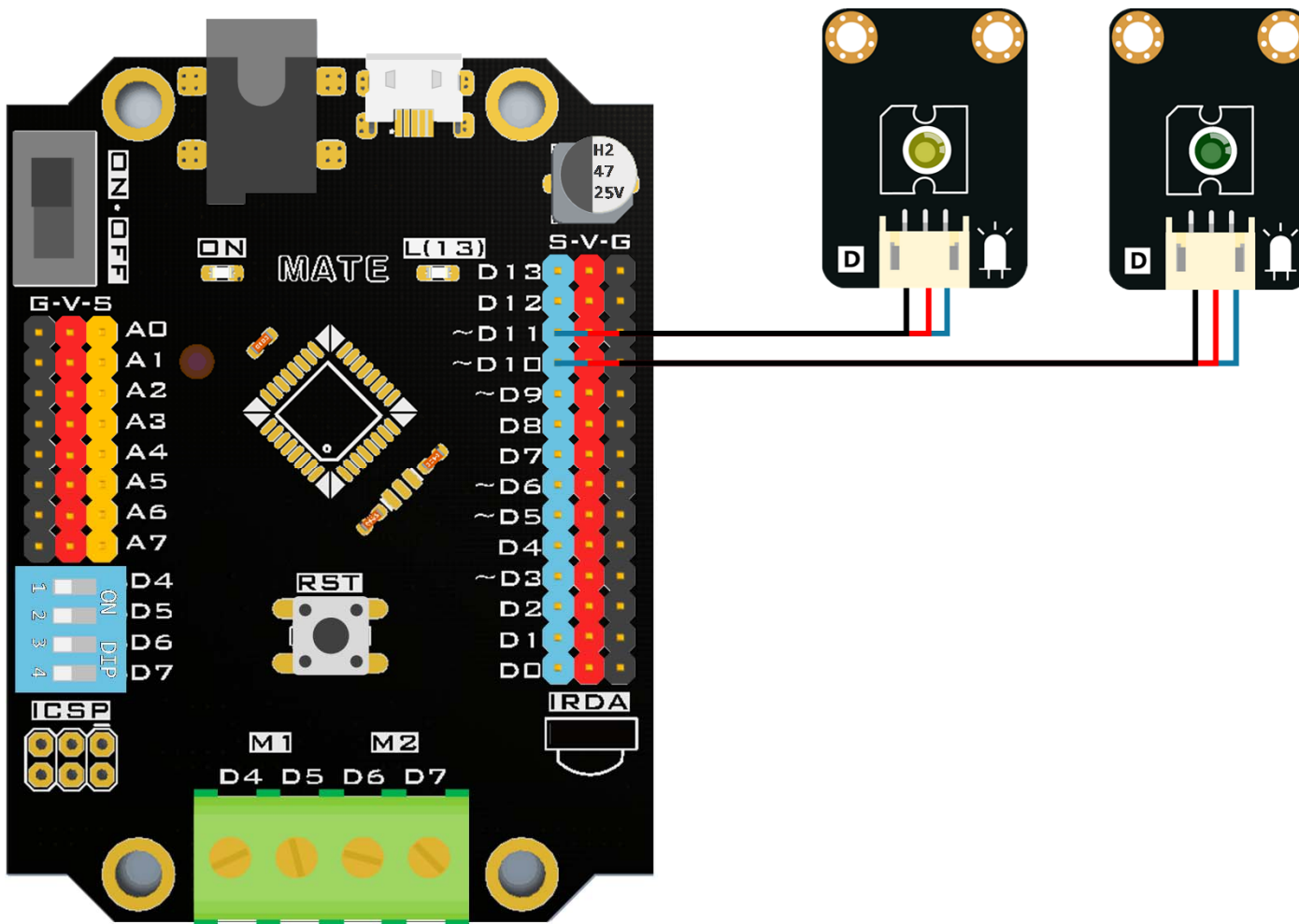
3

流光沙漏

可扩展任务 流光沙漏



硬件连接



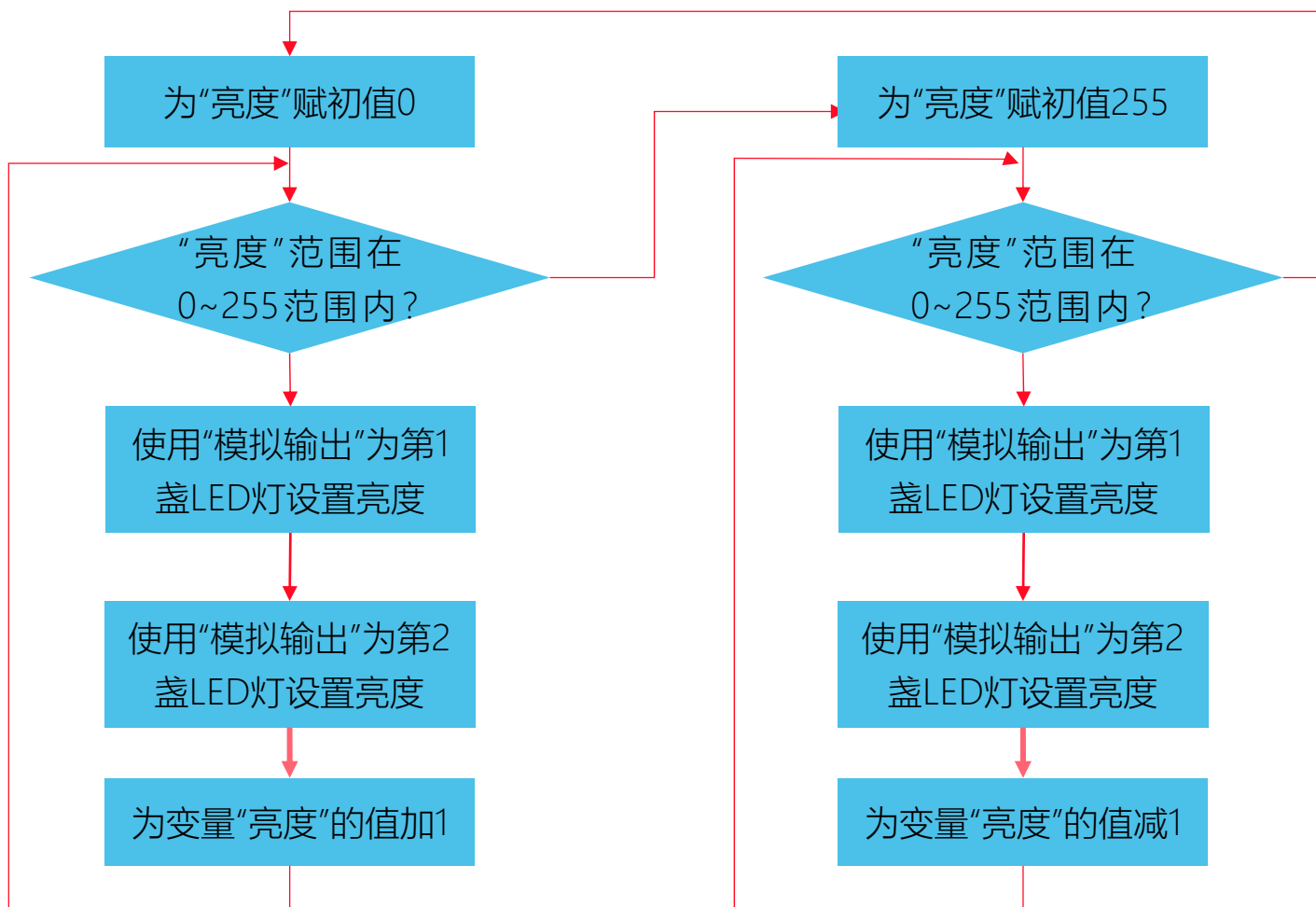
3

流光沙漏

可扩展任务 流光沙漏

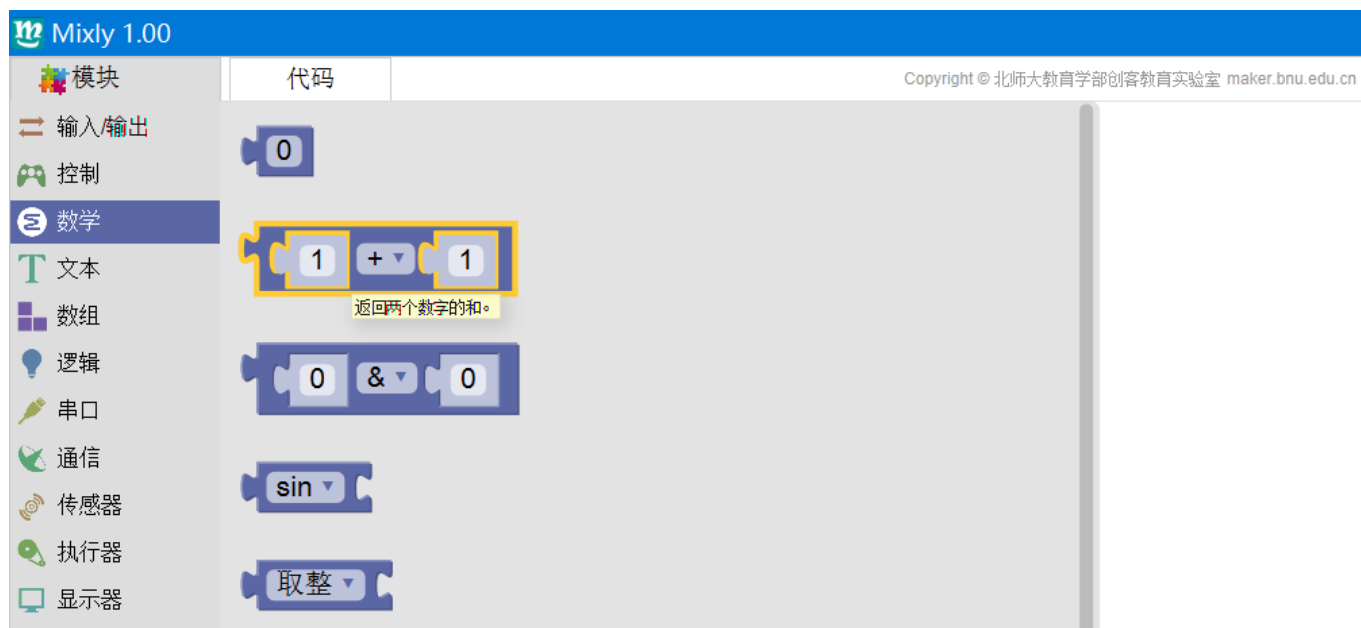


编程思路





认识新代码块



- “运算”代码块位于“数学”模块分类中。运算代码块支持9种运算：
加+、减-、乘×、除÷、取余数%和幂运算 a^b 。

3

流光沙漏

可扩展任务 流光沙漏



软件编写

```
使用 亮度 从 0 到 255 步长为 1  
执行  
  模拟输出 管脚# 10 赋值为 亮度  
  模拟输出 管脚# 11 赋值为 255 - 亮度  
  延时 毫秒 5  
使用 亮度 从 255 到 0 步长为 -1  
执行  
  模拟输出 管脚# 10 赋值为 亮度  
  模拟输出 管脚# 11 赋值为 255 - 亮度  
  延时 毫秒 5
```

3

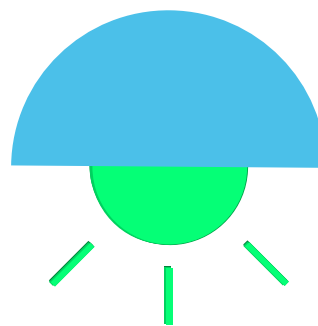
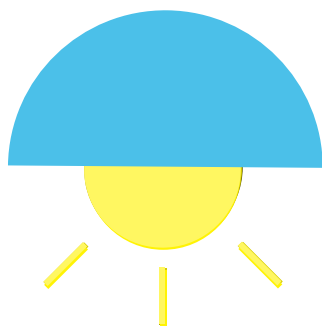
流光沙漏

可扩展任务 流光沙漏



拓展思考

- 沙漏运动的方向和速度各是由什么变量控制的?



3

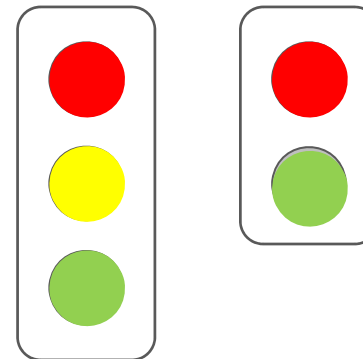
流光沙漏

自主扩展任务 可控交通灯



任务发布

- 使用五个OHG灯。其中三个红、黄、绿,代表汽车灯, 另外两个红、绿, 代表对向的行人灯, 例如: 东西向为汽车灯, 则南北向为行人灯。
- ①请参考第5课自主扩展任务的思路, 完成两组红绿灯程序的编写。
- ②加入行人干预: 只有当按下按钮+接在5号管脚时, 行人灯才会由红变绿。



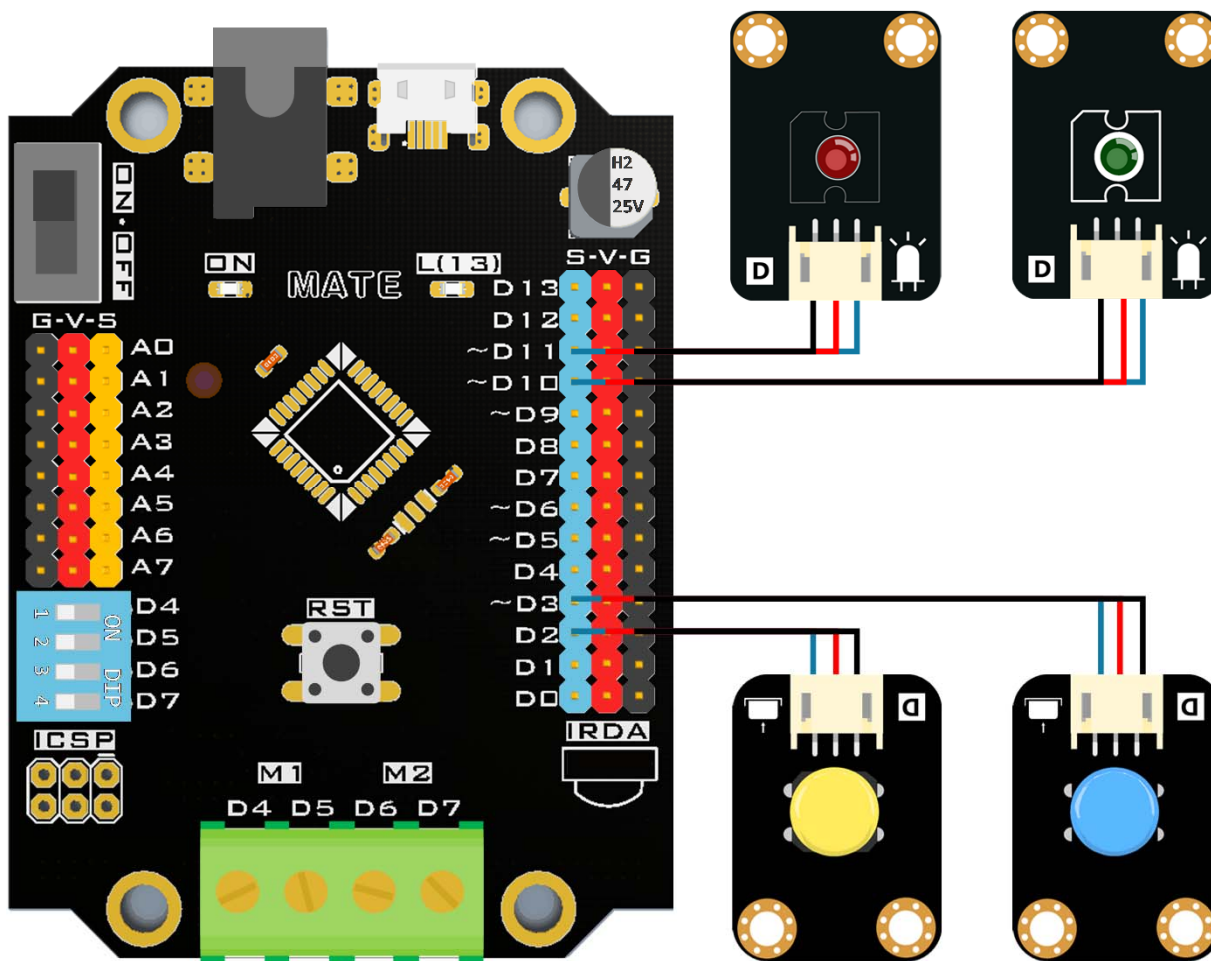
3

流光沙漏

自主扩展任务 可控交通灯



硬件连接



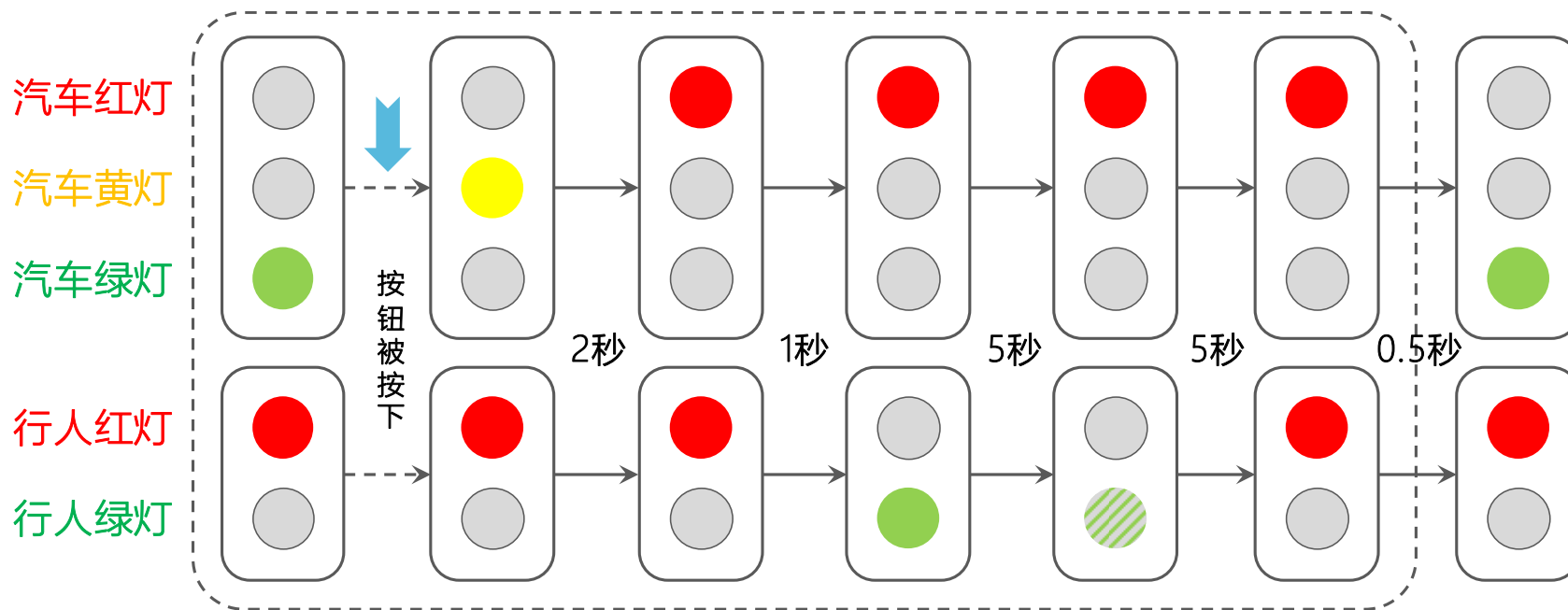
3

流光沙漏

自主扩展任务 可控交通灯



编程思路



3

流光沙漏

自主扩展任务 可控交通灯



软件编写

```
初始化  
声明 汽车红灯 为 整数 并赋值 11  
声明 汽车黄灯 为 整数 并赋值 10  
声明 汽车绿灯 为 整数 并赋值 9  
声明 行人红灯 为 整数 并赋值 8  
声明 行人绿灯 为 整数 并赋值 7  
数字输出 管脚# 汽车绿灯 设为 高  
数字输出 管脚# 行人红灯 设为 高
```

3

流光沙漏

自主扩展任务 可控交通灯



软件编写

```
if (digitalRead(PIN_2) == HIGH) {
  digitalWrite(LED_GREEN, LOW);
  digitalWrite(LED_YELLOW, HIGH);
  delay(2000);
  digitalWrite(LED_YELLOW, LOW);
  digitalWrite(LED_RED, HIGH);
  delay(1000);
  digitalWrite(LED_RED, LOW);
  digitalWrite(LED_GREEN, HIGH);
}
```

The code block is a Scratch-style 'if' statement. The condition is '如果' (If) with a gear icon, followed by '数字输入 管脚#' (Digital Input Pin #) set to '2'. The '执行' (Execute) block contains several steps: 1. '数字输出 管脚#' (Digital Output Pin #) '汽车绿灯' (Car Green Light) '设为' (Set to) '低' (Low). 2. '数字输出 管脚#' '汽车黄灯' (Car Yellow Light) '设为' '高' (High). 3. '延时' (Delay) '毫秒' (ms) '2000'. 4. '数字输出 管脚#' '汽车黄灯' '设为' '低'. 5. '数字输出 管脚#' '汽车红灯' (Car Red Light) '设为' '高'. 6. '延时' '毫秒' '1000'. 7. '数字输出 管脚#' '行人红灯' (Pedestrian Red Light) '设为' '低'. 8. '数字输出 管脚#' '行人绿灯' (Pedestrian Green Light) '设为' '高'.

3

流光沙漏

自主扩展任务 可控交通灯



软件编写

```
延时 毫秒 5000
使用 i 从 1 到 10 步长为 1
执行
  数字输出 管脚# 行人绿灯 设为 高
  延时 毫秒 250
  数字输出 管脚# 行人绿灯 设为 低
  延时 毫秒 250
数字输出 管脚# 行人红灯 设为 高
延时 毫秒 500
数字输出 管脚# 汽车红灯 设为 低
数字输出 管脚# 行人绿灯 设为 高
```

The code is written in a block-based programming language. It starts with a 5000ms delay block. This is followed by a loop block labeled '使用 i 从 1 到 10 步长为 1'. Inside the loop, there is a sequence of blocks: a '数字输出 管脚# 行人绿灯 设为 高' block, a 250ms delay block, a '数字输出 管脚# 行人绿灯 设为 低' block, and another 250ms delay block. After the loop, there is a '数字输出 管脚# 行人红灯 设为 高' block, followed by a 500ms delay block, a '数字输出 管脚# 汽车红灯 设为 低' block, and finally a '数字输出 管脚# 行人绿灯 设为 高' block.

3

流光沙漏

课后练习



1. 请使用按钮控制灯的亮度：按下按钮时灯的亮度不断变化，松开按钮时灯将维持在此时的亮度状态。
2. 请在1的基础上加入一个按钮，用来控制灯的关闭。