

NOVA 电子积木教程

HD 版

(支持 Nduino HD)

STARLAB 创客社区

好好搭搭在线

目录

第一部分 玩转硬件

| | |
|--------------------------|----|
| 第一节 初识体验 NOVA 电子积木 | 6 |
| 示例 1-1：点亮 LED | 12 |
| 第二节 数字量输出及其控制 | 21 |
| 示例 2-1：LED 的闪烁 | 21 |
| 示例 2-2：交通灯（红绿灯） | 23 |
| 第三节 数字显示屏 | 25 |
| 示例 3-1：用数码管显示数字 | 26 |
| 第四节 模拟量输入与采集 | 28 |
| 示例 4-1：环境光检测仪 | 30 |
| 示例 4-2：超声波测距仪 | 32 |
| 示例 4-3：温湿度计 | 33 |
| 第五节 模拟量的输出 | 35 |
| 示例 5-1：LED 调光 | 35 |
| 示例 5-2：旋钮调光 | 39 |
| 第六节 电机的驱动 | 41 |
| 示例 6-1：可调速电风扇 | 41 |
| 示例 6-2：双电机驱动 | 42 |
| 第七节 蜂鸣器 | 44 |
| 示例 7-1：两只老虎 | 44 |
| 第八节 RGB 七彩灯 | 47 |
| 示例 8-1：RGB 交通灯 | 48 |

第二部分 编程入门

| | |
|--|----|
| 第九节 编程基础知识介绍 | 51 |
| 第十节 数字量输入与条件判断指令 | 53 |
| 示例 10-1：用数码管显示开关量传感器的状态，以按钮开关为例。 | 54 |
| 示例 10-2：门铃 | 56 |
| 第十一节 变量 | 57 |
| 第十二节 数学与逻辑运算 | 59 |

| | |
|-----------------------------|----|
| 示例 12-1：华氏温度计 | 60 |
| 示例 12-2：光控灯（光敏开关版） | 61 |
| 示例 12-3：夜间声控灯 | 63 |
| 第十三节 随机数 | 65 |
| 示例 13-1：摇号机（基本版） | 65 |
| 第十四节 串口打印 | 68 |
| 示例 14-1：串口打印单个传感器数值 | 70 |
| 示例 14-2：串口打印多个传感器数值 | 72 |
| 第十五节 新建功能块 | 74 |
| 示例 15-1：RGB 交通灯（功能块版） | 75 |

第三部分 硬件进阶

| | |
|------------------------------|-----|
| 第十六节 舵机控制 | 79 |
| 示例 16-1：舵机摆动 | 80 |
| 第十七节 红外遥控 | 81 |
| 示例 17-1：红外遥控 LED | 82 |
| 示例 17-2：用数码管显示红外遥控器的键值 | 83 |
| 第十八节 MP3 音乐播放 | 86 |
| 示例 18-1：红外遥控 MP3 播放器 | 88 |
| 第十九节 炫酷点阵屏 | 90 |
| 示例 19-1：按坐标点亮点阵模块 | 91 |
| 示例 19-2：音乐动感点阵屏——柱状图 | 92 |
| 第二十节 RTC 时钟模块 | 95 |
| 示例 20-1：简易时钟 | 96 |
| 第二十一节 蓝牙远程控制 | 101 |
| 示例 21-1：蓝牙控制 LED | 107 |

第四部分 软件进阶

| | |
|---|-----|
| 第二十二节 计时器 | 116 |
| 示例 22-1：长按点亮 LED，松开熄灭 LED | 116 |
| 示例 22-2：LED 闪烁的同时，用电位器控制风扇的转速（等待指令） | 117 |
| 示例 22-3：LED 闪烁的同时，用电位器控制风扇的转速（系统运行时间） | 119 |

| | |
|--------------------------------------|-----|
| 第二十三节 输入计数及其应用 | 122 |
| 示例 23-1：记录按键按下松开的次数，并显示到数码管上 | 122 |
| 示例 23-2：点控 LED | 124 |
| 示例 23-3：按键控制切换 RGB 灯的颜色 | 126 |
| 示例 23-4：亮度传感器控制 LED 灯的亮灭（点控） | 127 |
| 示例 23-5：用两个按键控制 MP3 的上一曲/下一曲切换 | 130 |
| 第二十四节 数组 | 134 |
| 示例 24-1：抽签机随机取数不重复 | 135 |
| 第五部分 机器人应用 | |
| 第二十五节 智能小车的运动 | 143 |
| 示例 25-1：智能小车的前进和后退 | 143 |
| 示例 25-2：智能小车的左转和右转 | 145 |
| 第二十六节 红外遥控智能小车 | 146 |
| 示例 26-1：红外遥控智能小车 | 146 |
| 第二十七节 智能小车实现自动避障 | 148 |
| 示例 27-1：自动避障智能小车 | 148 |
| 第二十八节 自动循线智能小车 | 150 |
| 示例 28-1：将灰度传感器的值显示到串口助手中去 | 150 |
| 示例 28-2：智能小车的自动循线 | 153 |
| 附录一 百变小魔盒拼装说明书 | 155 |
| (1) 安装基础框架 | 155 |
| (2) 结构案例 | 158 |
| 附录二 智能小车拼装说明书 | 170 |

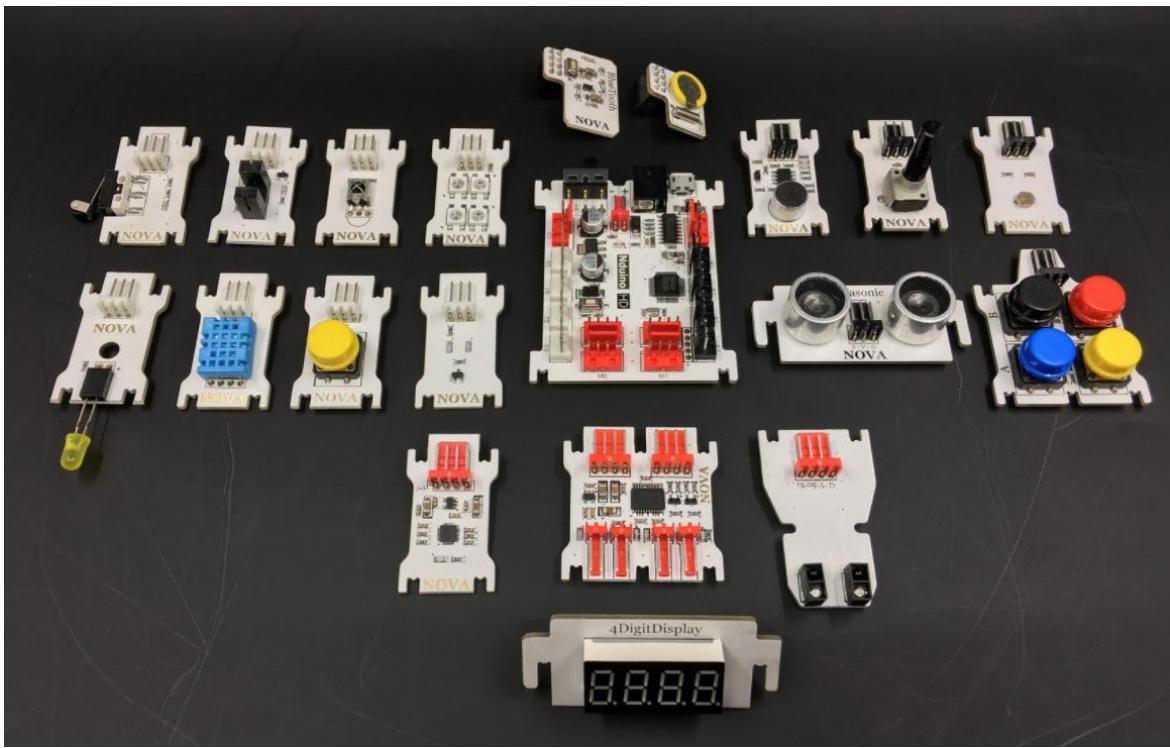
第一部分

硬件入门

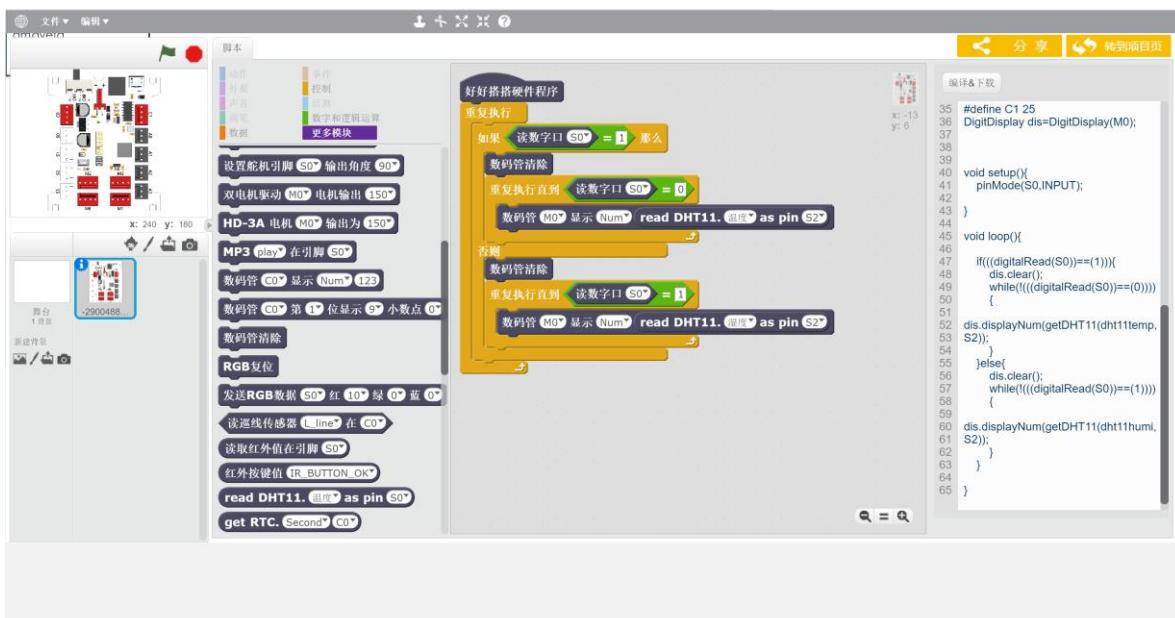
第一节 初识体验 NOVA 电子积木

一、认识 NOVA 电子积木

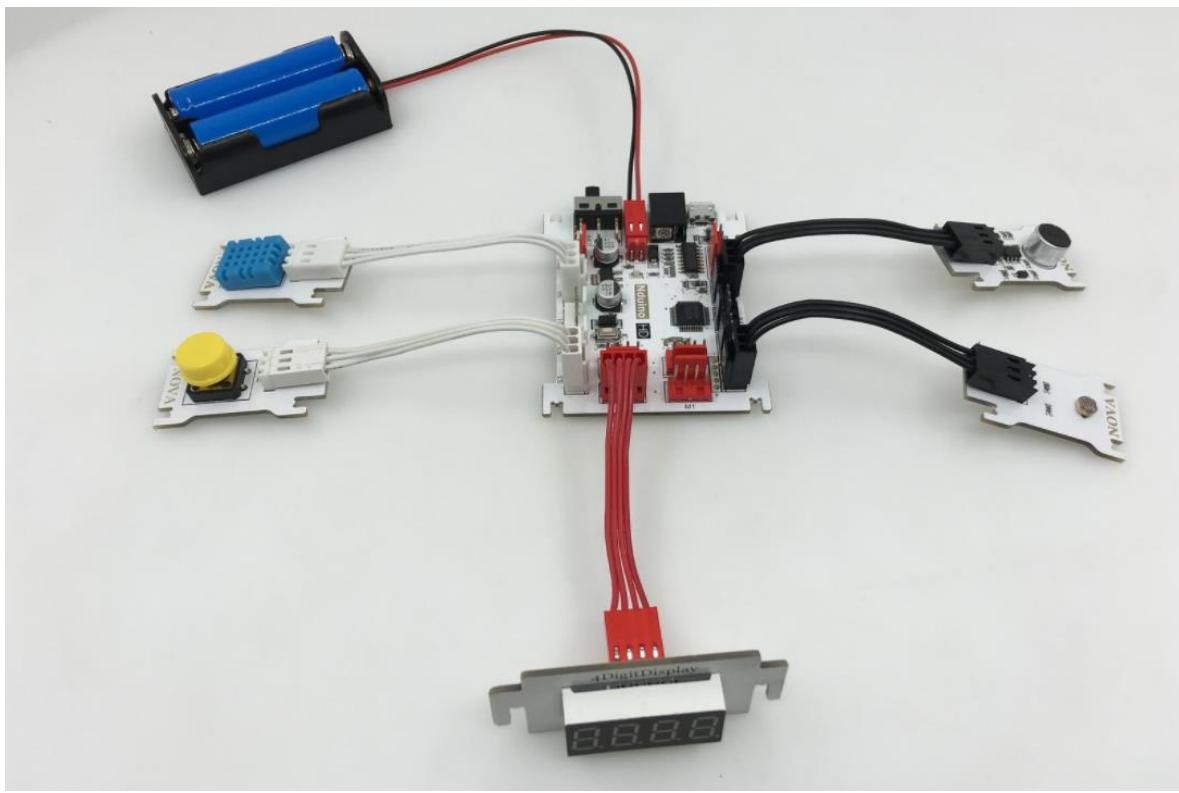
NOVA 电子积木是在 Arduino 基础上开发的一系列可编程开源硬件模块，包括主控模块 Nduino，和多种输出模块、传感器模块、通信模块等功能模块。



采用好好搭搭在线图形化云编译平台进行编程，并为其开发了专用的 NOVA 系列图形化指令，进一步降低编程难度，方便实现更加多样化的创造。



硬件连接非常简单：两个相同颜色的接口相互连接。



二、首次使用好好搭搭在线编程平台

1. 登入好好搭搭在线编程平台，网址:www.haohaodada.com

首页 创作 发现 学习 资源

登录 加入

2016年贵州省中小学生“多彩创客、智造无限”
创意编程及开源电子设计类大赛活动专区

MYF 猫友汇 Scratch Maker CLUB haohaodada 猫友汇-好好搭搭
创客教育云讲堂

创意编程

查看全部

超级玛丽

作者: 创意编程
988 ★ 573 0 243603

Epic Ninja

作者: 创意编程
823 ★ 601 0 157787

接鸡蛋1.0

作者: 创意编程
212 ★ 107 0 65250

魂斗罗 V1.2外挂版

作者: 创意编程
151 ★ 98 0 63057

恭喜通关！！

作者: 创意编程
135 ★ 85 0 54646

精华作品

查看全部

WELCOME young mario

作者: 贵州省实验小学
157787

忍者龟

作者: 创意编程
823 ★ 601 0 157787

红魔

作者: 创意编程
151 ★ 98 0 63057

气球

作者: 创意编程
135 ★ 85 0 54646

2. 注册账号:



注册账号点击“加入”按钮



输入用户名、密码并确认密码，邀请码可不填。点击“注册”按钮，会弹出注册成功提示。点击“退出”按钮关闭注册窗口。

3. 登录账号:



点击右上角“登录”按钮，弹出用户登录窗口，输入已注册好的用户名和密码，点击“登录”按钮，即可登录。



登录成功后，右上角会显示用户名。

4. 下载谷歌浏览器（如果已安装 chrome、360 浏览器、猎豹浏览器，忽略此步）



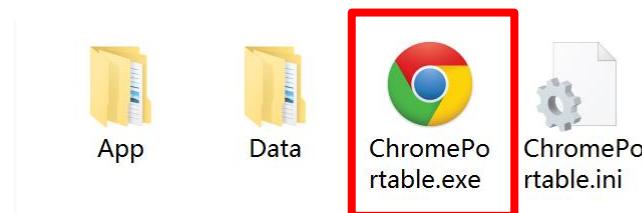
点击“资源”按钮



点击“谷歌浏览器绿色版下载”。



点击下载，下载完成后解压压缩包



双击“ChromePortable.exe”文件，启动谷歌浏览器。

5. 下载好好搭搭编程插件：



The screenshot shows the haohaodada.com website's resource download page. It features a green header with tabs for 首页 (Home), 创作 (Create), 发现 (Discover), 学习 (Learn), and 资源 (Resources). Below the header, a message says '请使用谷歌浏览器体验更佳' (Best experience with Google Chrome). The main content area is titled '资源下载' (Resource Download) and lists several items with their download links and dates. One item, '好好搭搭插件下载' (Good Good Dada Plugin Download) from March 27, 2016, is highlighted with a red box and a red arrow pointing to it.

资源界面下，点击“好好搭搭插件下载”。

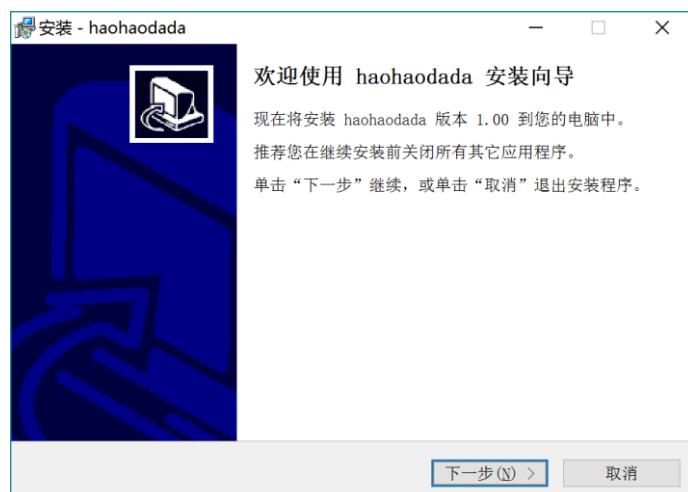


This screenshot shows the '好好搭搭插件下载' (Good Good Dada Plugin Download) page. It has a green header with tabs for Home, Create, Discover, Learn, and Resources. The main content area is titled '[资源下载] - [软件资源]' (Resource Download - Software Resource) and '好好搭搭插件下载'. It includes a date (2016-03-27 10:53:53), a reading count (阅读16851), and a sharing button ('分享到QQ'). A red box and a red arrow point to the '点击下载' (Click to Download) button.

点击“点击下载”字样，下载好好搭搭插件

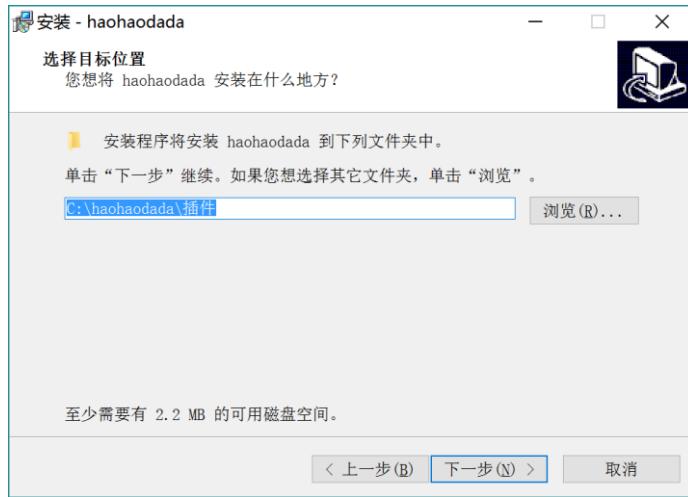


下载文件为“haohaodada_setup.exe”，双击安装。

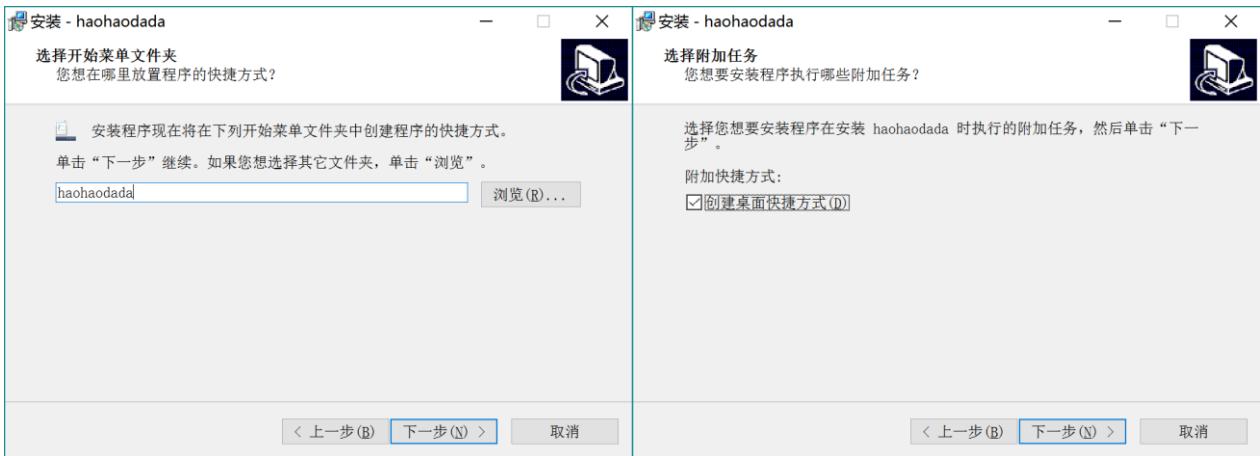


This screenshot shows the first step of the 'haohaodada' setup wizard. The window title is '安装 - haohaodada'. The main content area is titled '欢迎使用 haohaodada 安装向导' (Welcome to the haohaodada Installation Wizard). It contains the text '现在将安装 haohaodada 版本 1.00 到您的电脑中。推荐您在继续安装前关闭所有其它应用程序。单击“下一步”继续, 或单击“取消”退出安装程序。' (Now installing haohaodada version 1.00 to your computer. It is recommended to close all other applications before continuing. Click 'Next' to continue, or click 'Cancel' to exit the installation program.). At the bottom, there are '下一步 (N) >' and '取消' (Cancel) buttons.

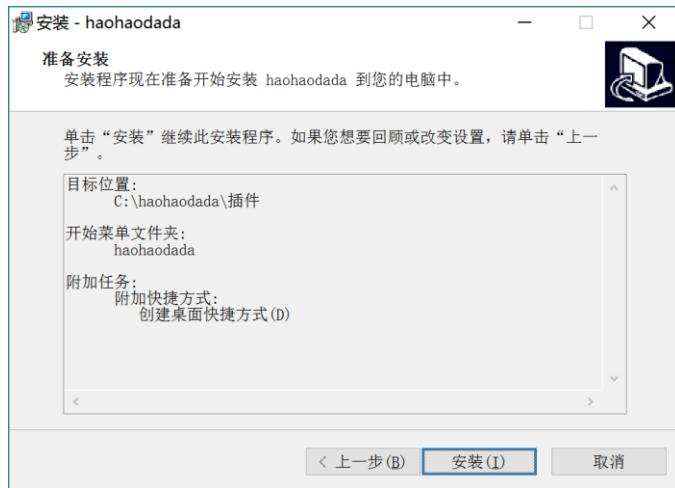
点击“下一步”。



点击“浏览”选择插件安装路径，点击“下一步”继续。



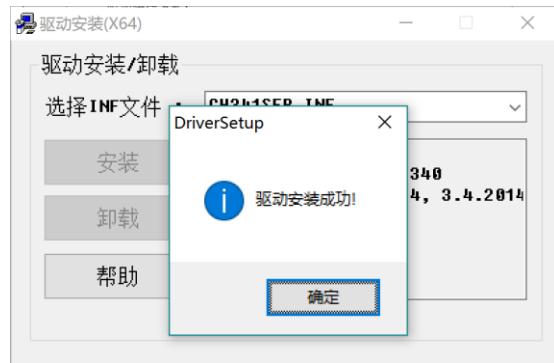
点击“下一步”。



点击“安装”。



点击“安装”。



必须等到“驱动安装成功！”提示框弹出后，方可关闭窗口。

好好搭搭插件程序安装完成！

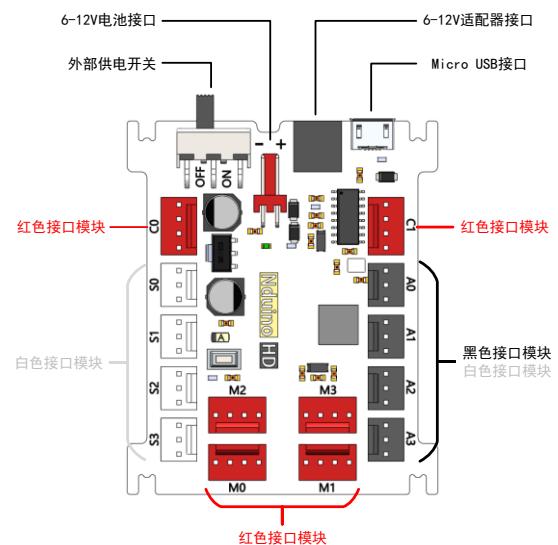
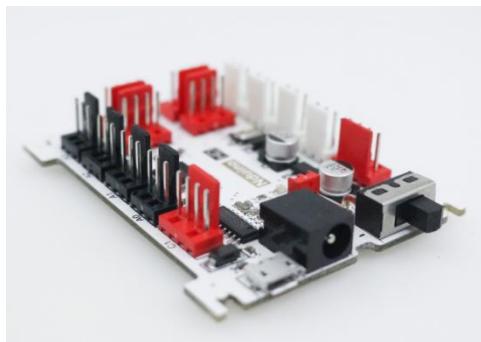
接下来用一个案例，亲身体验 NOVA 电子积木的使用。

示例 1-1：点亮 LED

用 NOVA 电子积木点亮一盏 LED。

认识新模块：

Nduino HD 主控板：



模块连接规则：相同颜色的接头互联

- (1) 白色接口模块可以连接 Nduino 的白色接口 S0-S3，也可以连接主控板的黑色接口 A0-A3；
- (2) 黑色接口模块只能连接 Nduino 的黑色接口 A0-A3。
- (3) 红色接口模块连接 Nduino 的红色接口。

供电方案：

- (1) Micro USB 连接可以为 Nduino 提供 5V 电压和最大 500mA 电流，当 Nduino 没有连接红色接口模块时，仅由 USB 供电也可正常工作。
- (2) 当连接红色接口模块时，需由电源适配器（9V1A）或外接电池（电压 6V 以上）供电，Nduino 才能正常工作。



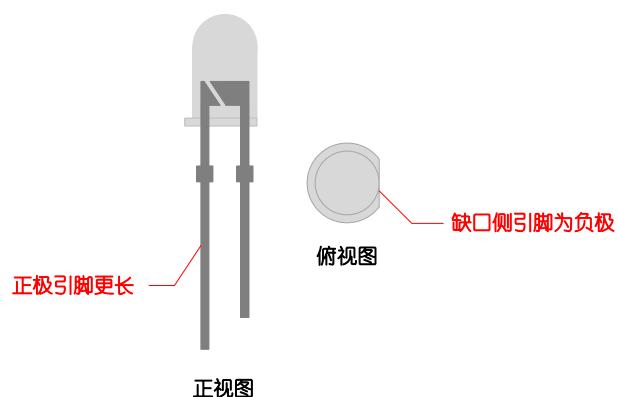
9V1A 电源适配器



磷酸铁锂电池

注意：磷酸铁锂电池 1 节 3.2V，切勿将其用于家用电器和其他电子设备！

LED 发光二极管：由含镓 (Ga)、砷 (As)、磷 (P)、氮 (N) 等的化合物制成。具有单向导通性，即 LED 有电流正向流入能点亮，反向流入或无电流则不亮。辨别发光二极管的正负极方法如右图：



LED 模块：用于连接驱动 LED 发光二极管



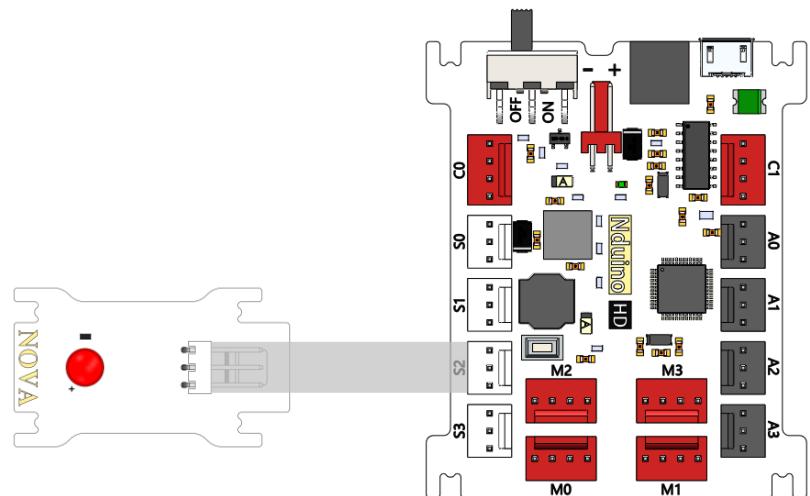
元器件列表:

Nduino HD 主控板 ×1

LED 模块（红） ×1

3Pin 2510 连接线（白） ×1

电路连接:



进入 NOVA 编程界面：



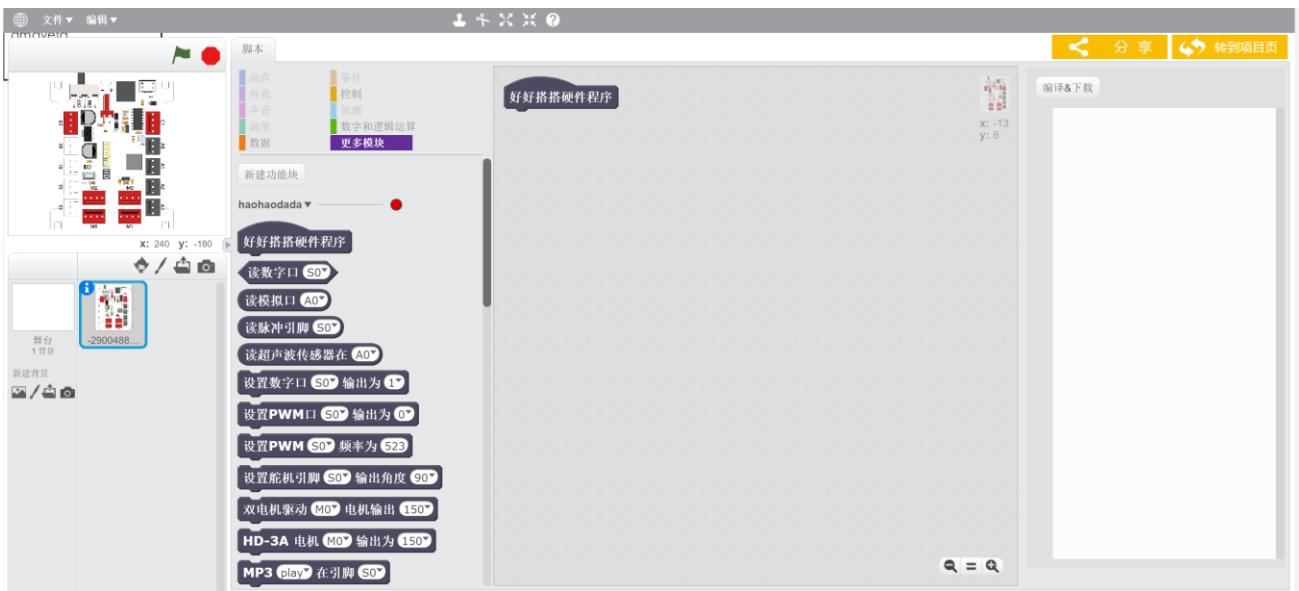
点击“创作”按钮



选择“Haohaodada_NOVA”点击进入



点击“转到设计页”，进入编程界面



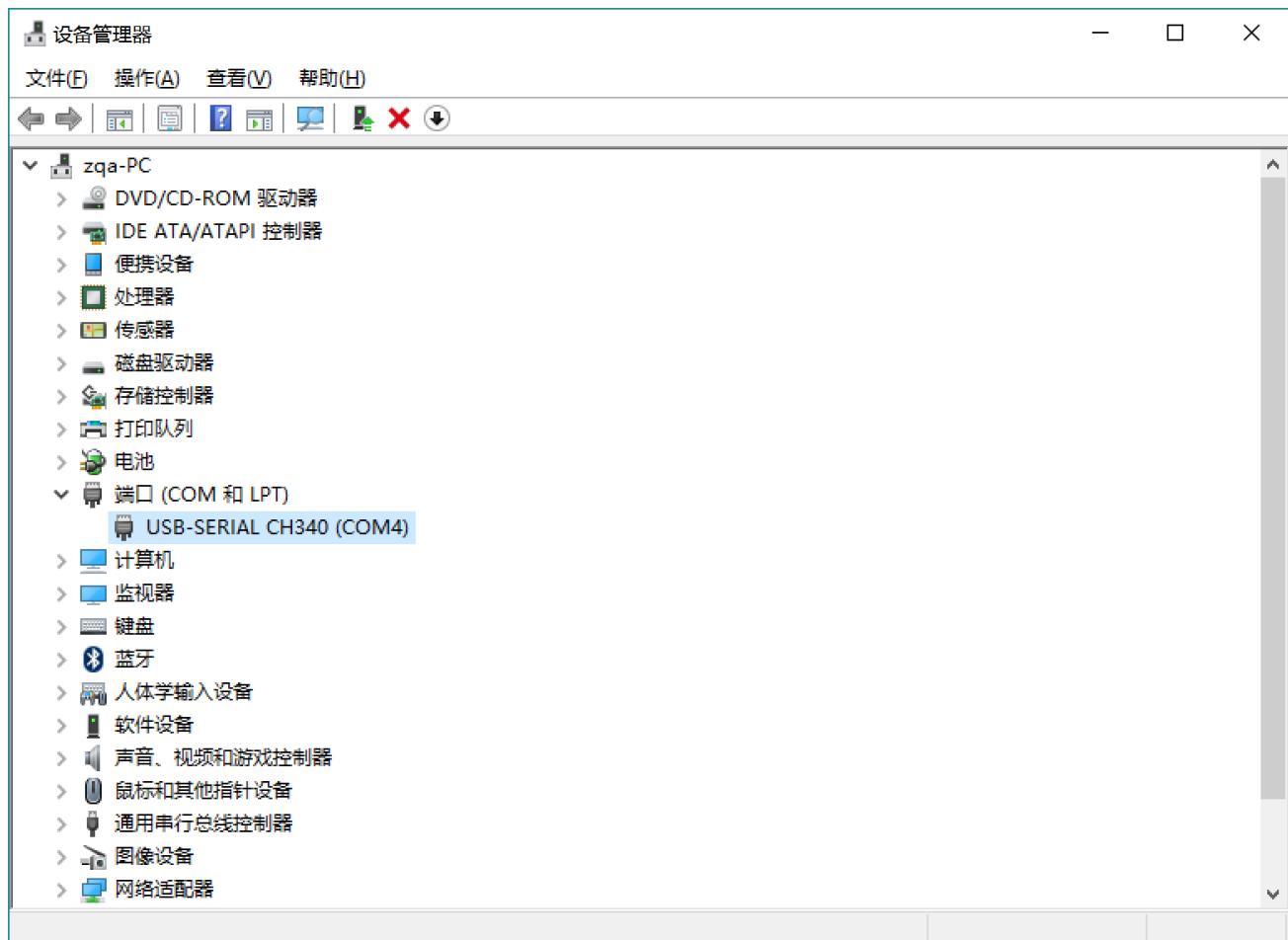
接下来开始编程吧！

打开硬件下载插件

在好好搭搭在线编程平台上进行硬件编程，都需要打开“硬件下载插件”。



“选择串口”出现“COM 口”，且 COM 口的编号与“设备管理器”中一致，则表示驱动程序安装正常。



注意：打开“硬件下载插件”后，一定不能关闭，否则程序无法下载到主控板 Nduino HD 中。

编写第一个程序——点亮 LED 灯

LED 灯的亮和灭两种状态，自然地联想到开和关两种状态，进一步联想到 0 和 1 两种状态，再进一步对应上“无”和“有”两种状态。像这类物理量被称为开关量，是一种数字量。

所以需要使用数字量相关指令，对于主控板 Nduino HD 来说，点亮 LED 是由 Nduino 输出信号给 LED，因此使用“设置数字口...输出为...”指令：



设置端口

点击“S0”，弹出下拉菜单，选择数字口：



选择端口的原则是：**与硬件连接一致！**本例中 LED 模块是连接到 Nduino HD 的 S3 端口，所以指令选择“S3”。

输出可选择“0”或“1”，对应上“开”和“关”，对应上 LED 是“亮”和“灭”。



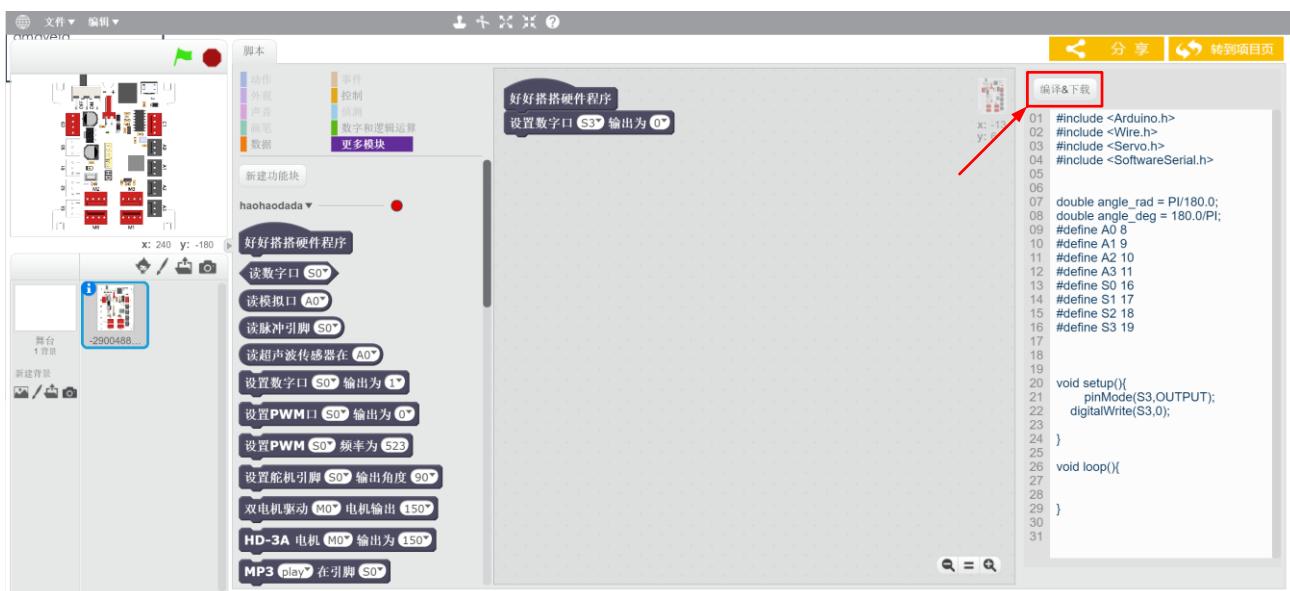
注意：具体“0”对应的是“开”还是“关”，LED 是“亮”还是“灭”，不一定！不同的硬件有着不同的标准，所以大家不应该养成“0”对应“关”和“1”对应“开”这样的定式思维！

Nduino HD 的标准是“0”对应的是点亮 LED，“1”对应的熄灭 LED。

那么本例中点亮 LED 的程序为：



编译和下载程序



点击右上角“编译&下载”按钮。



点击“下载”按钮，将程序下载到 Nduino HD 中。

再次提示：“硬件下载插件”在整个过程中不能关闭！

完成！可以看到 LED 发光二极管亮起。

NOVA 电子积木完整使用流程：

搭建电路
连接电脑



打开
硬件下载插件



登入
好好搭搭在线

<http://www.haohaodada.com/show.php?id=69015>

编写程序



编译&下载
程序

本教程后续内容仅对“搭建电路”和“编写程序”做详细说明。

第二节 数字量输出及其控制

上节课大家已经掌握了 NOVA 电子积木的电路连接和 Mixly 软件的基本操作，并实现了点亮 LED 的程序。本次课程用几个 LED 的案例让大家理解什么是数字量。

示例 2-1：LED 的闪烁

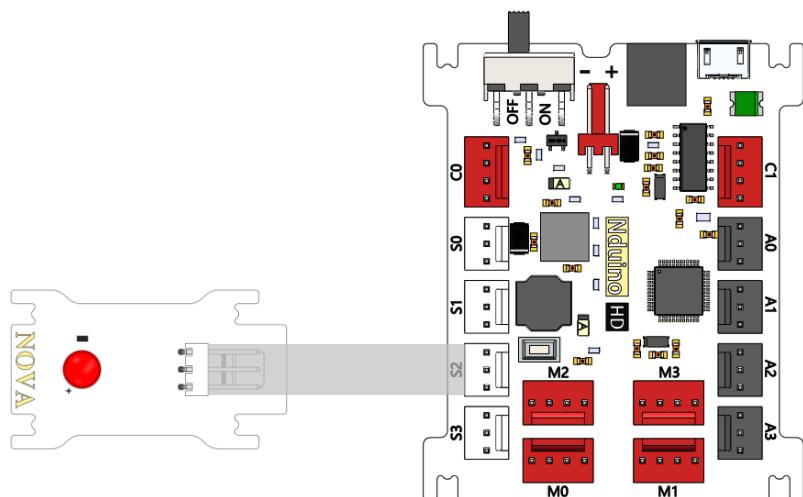
元器件列表：

Nduino HD 主控板 ×1

LED 模块（红） ×1

3Pin 2510 连接线（白） ×1

电路连接：



认识新指令——等待指令

等待 1 秒

等待指令的作用是让程序暂停执行一段时间，时间的长短可以设置。

认识新指令——重复执行指令

重复执行



重复执行指令的作用是让其内部的程序不断重复的运行，永无止境。在好好搭搭硬件编程中，重复执行指令内部的程序被称为主程序。

程序分析:

我们来仔细思考下“闪烁”这个词，闪烁是灯亮一段时间，之后灭一段时间，如此循环往复。那么，“闪烁”应该被拆分成“亮”、“灭”、“持续一段时间”这些动作的组合。其中，“亮”和“灭”可以通过“设置数字口”指令实现：



“持续一段时间”需要使用等待指令。

不断的“闪烁”需要使用重复执行指令。

程序编写:



拓展项目一：SOS 救援信号

背景知识：

S.O.S 是国际摩尔斯电码救难信号，并非任何单字的缩写。国际无线电报公约组织于 1908 年正式将它确定为国际通用海难求救信号。它的电码为“...---...”（三短三长三短）在电报中是发报方最容易发出，收报方最容易辨识的电码。

项目要求：

实现一个 LED 的闪烁步骤如下：

- (1) 连续短闪烁三次，单次闪烁时间间隔为 0.3 秒
- (2) 连续长闪烁三次，单次闪烁时间间隔为 1 秒
- (3) 连续短闪烁三次，单次闪烁时间间隔为 0.3 秒

两个周期之间的时间间隔为 3 秒。

示例 2-2：交通灯（红绿灯）

元器件列表：

Nduino HD 主控板 ×1

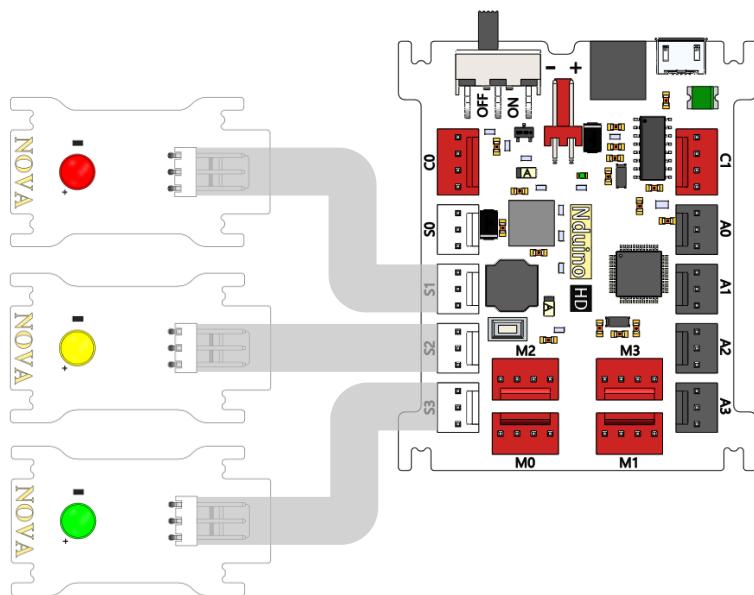
LED 模块（红） ×1

LED 模块（黄） ×1

LED 模块（绿） ×1

3Pin 2510 连接线（白） ×3

电路连接：



程序分析：

同学们可以去仔细观察或回想一下交通路口的红绿灯，它们是如何运作的：红黄绿三盏灯依次亮灭，各持续一段时间。程序的编写要点是用不同的端口控制不同的 LED 模块，实现依次亮灭的目的。

程序编写：

初始化程序：将三支 LED 灯全部关灭。

初始化程序是在运行开始时只执行一次的程序，位于“好好搭搭硬件程序”之后，重复执行指令之前。

好好搭搭硬件程序

设置数字口 S0 输出为 1

设置数字口 S2 输出为 1

设置数字口 S3 输出为 1

初始化程序：关灭三个LED

重复执行

设置数字口 S3 输出为 0 点亮绿色LED

等待 8 秒 绿色LED亮8秒

设置数字口 S3 输出为 1 关灭绿色LED

设置数字口 S2 输出为 0 点亮黄色LED

等待 3 秒 黄色LED亮3秒

设置数字口 S2 输出为 1 关灭黄色LED

设置数字口 S0 输出为 0 点亮红色LED

等待 8 秒 红色LED亮3秒

设置数字口 S0 输出为 1 关灭红色LED

主程序

第三节 数字显示屏

右图这种计算器相信大家都不陌生，这种液晶显示屏只能显示数字，简称为数码管。相对于电视或手机屏幕，这种显示屏使用上更加简单，当创客项目中仅需要显示数字而不需要显示文字、图片信息时，可以使用数码管来简单快速的实现。



NOVA 电子积木中提供了两种数码管，一种数字和小数点，另一种显示时间。

认识数码管



显示数字数码管



显示时间数码管

显示数字数码管：有小数点位，可以显示小数

显示时间数码管：有时钟冒号，适合显示时刻

认识新指令——整体数字显示指令



整体数字显示指令可以方便的将四位以内的数字显示到数码管上。显示整数选择“Num”模式，显示带小数位的数字（浮点数）选择“Float”模式。

认识新指令——数码管清除指令



数码管清除指令用于将数码管的显示全部清除，即将数码管上所有的 LED 关灭。

认识新指令——按位显示指令



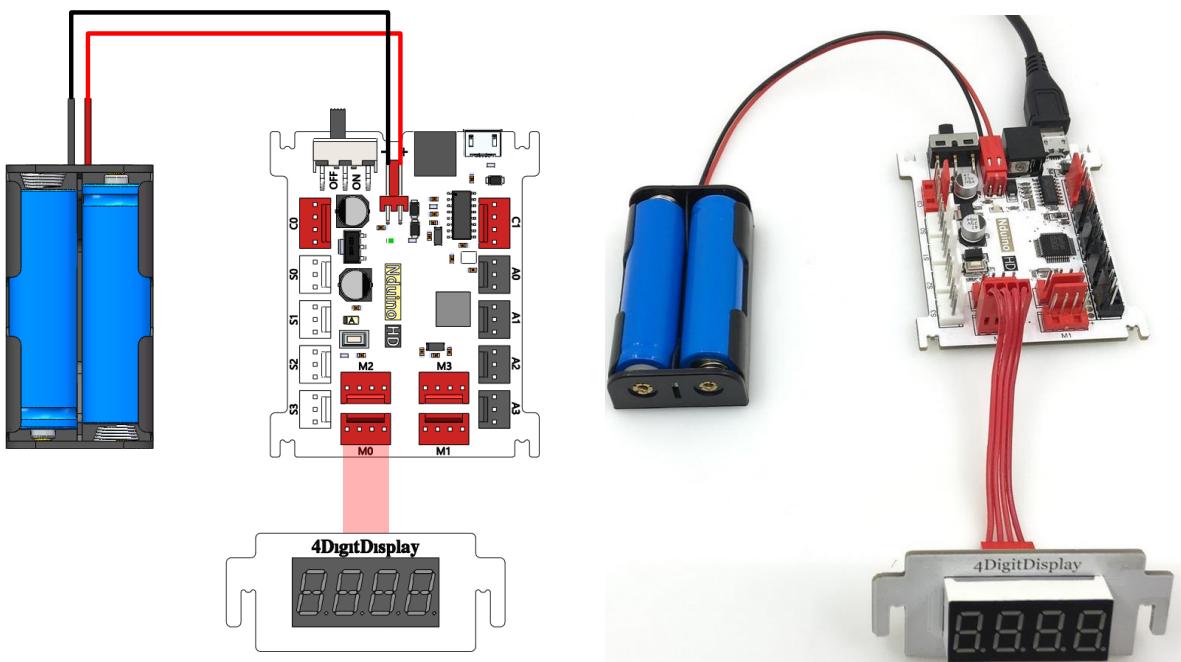
按位显示指令可以指定四位数码管中的某一位进行显示。

示例 3-1：用数码管显示数字

元器件列表：

1. Nduino HD 主控板×1
2. 数码管模块（数字）×1
3. 数码管模块（时间）×1
4. 4Pin 2510 连接线（红）×2

电路连接：

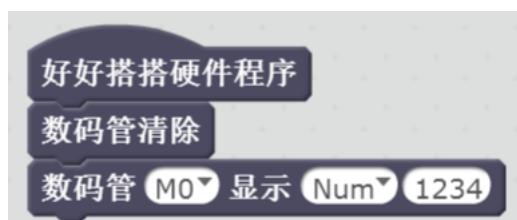


元器件列表：

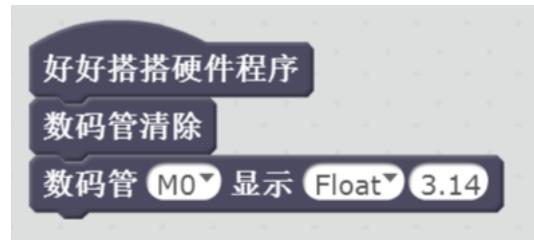
1. Nduino HD 主控板 × 1
2. 数码管模块 × 1
3. 4Pin 2510 连接线（红）× 1

Haohaodada 程序编写：

显示一个整数：



显示一个小数：



显示一个变量：



第四节 模拟量输入与采集

本次课将带大家认识传感器，传感器是一种检测装置，能感受到被测量的信息，并能将感受到的信息，按一定规律变换成为电信号或其他所需形式的信息输出，以满足信息的传输、处理、存储、显示、记录和控制等要求。

从传感器输入的数值看，一种是只有两个取值“0”和“1”的传感器，称为数字量传感器；另一种是有一个取值范围，如0到100、0到1023，称为模拟量传感器。

数字量传感器

详见第十课《数字量输入与条件判断指令》

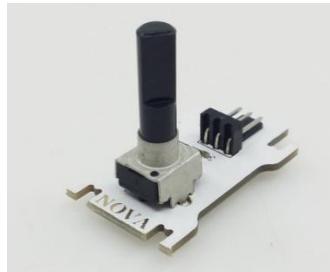
模拟量接口传感器

模拟量是指在一定范围连续变化的量，比如温度、亮度、距离、重量等。

模拟量传感器就是采集这类物理量，并将其转换为电压或电流信号的传感器。

认识新模块和新指令：

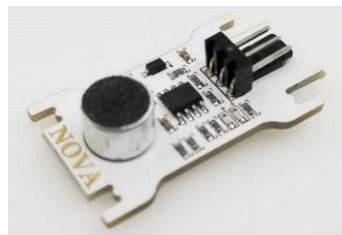
电位计模块：一种旋转变阻器，可以用作旋钮，可以感知角度的变化。



光敏传感器模块：用来检测当前的光照强度。



声音传感器模块：用来检测当前环境的声音强度。



以上三种模块都是模拟量传感器模块，输送到主控板中的是电压值，可以直接用“读模拟量”指令读取：

读模拟口 A0▼

主控板上能读取模拟量传感器的接口是黑色接口 A0~A3。

另外还有一类传感器，它测量的是模拟量如距离、温度、湿度，但是输送给主控板的信号不是电压值，所以不能用读模拟口指令读取数值。好好搭搭为这类传感器设计了专用的指令。

超声波测距传感器：通过发射接收超声波来测量超声波模块与障碍物距离。超声波测距传感器共设计够两种，图片和对应的指令如下：



读超声波传感器在 A0▼

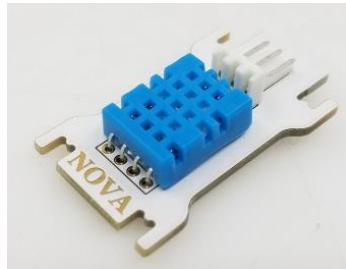
V1.0 版本

读2.0超声波传感器在 S0▼

V2.0 版本

温湿度传感器模块：用来检测当前环境的温度和湿度。下图中的蓝色塑料块即是温湿度传感器的核心器件——DHT11，所以对应的指令名称为“读 DHT11”。

注意不能直接放入水中测量水温！



读 DHT11. 温度▼ 在引脚 S0▼

用这些传感器配合数码管可以制作各式各样的测量仪表，如超声波测距仪、光照强度仪、噪音计

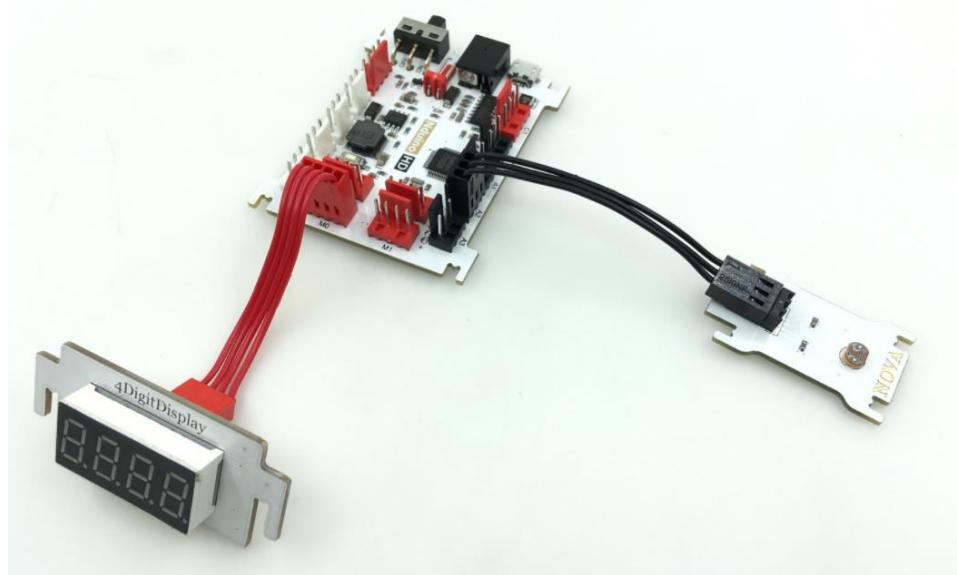
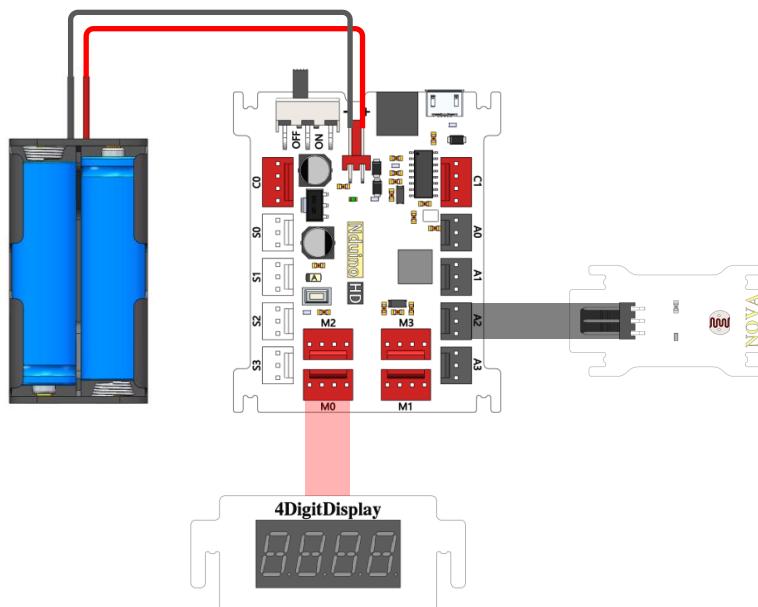
示例 4-1：环境光检测仪

用数码管实时的显示环境光的数值

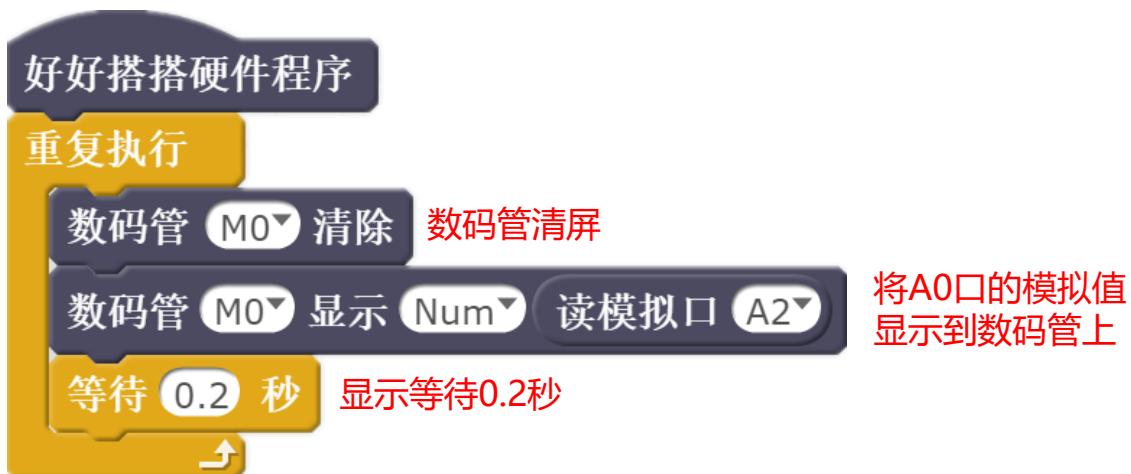
元器件列表：

1. Nduino HD 主控板×1
2. 亮度传感器模块×1
3. 数码管模块（计数）×1
4. 3Pin 2510 连接线（黑）×1
5. 4Pin 2510 连接线（红）×1

电路连接：



程序编写：



以上程序即可实现环境光的检测，同学们试着测量各种情况下的亮度值。

| 环境情况 | 显示数值 |
|---------|------|
| 白天房间里 | |
| 用阴影遮挡光敏 | |
| 直接盖在光敏上 | |
| 在灯光下 | |
| 在阳光下 | |

将传感器换成声音传感器，程序不变，即可制作环境声检测仪。同样，同学们也试着测量下各种环境中的声音响度值吧

| 环境情况 | 显示数值 |
|----------|------|
| 比较安静的房间里 | |
| 2人正常对话 | |
| 大声喊叫 | |
| 教室里 | |
| 上体育课的操场上 | |

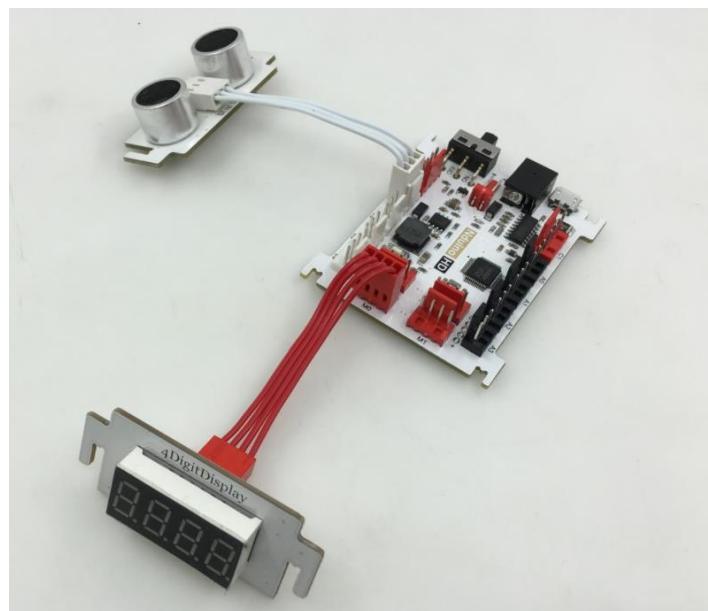
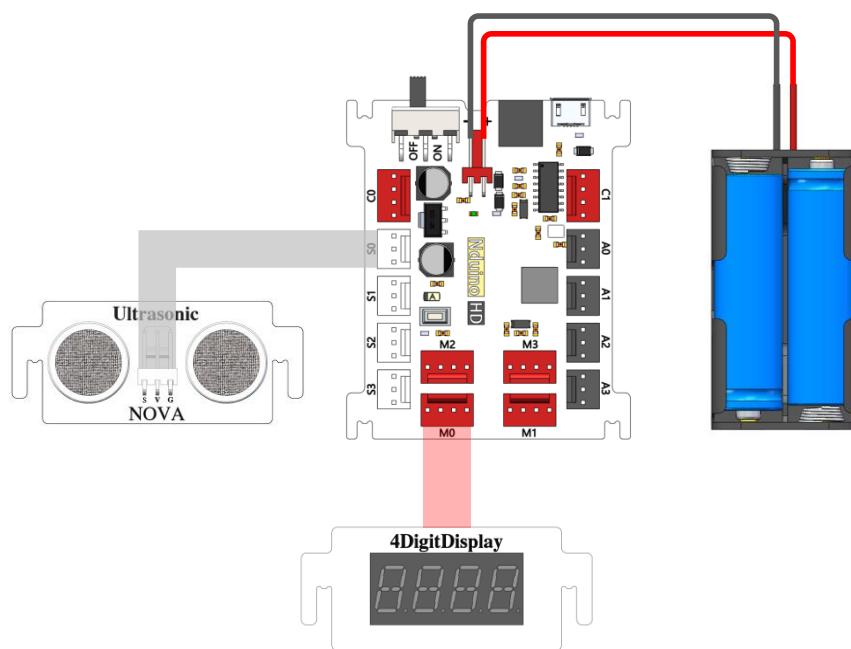
示例 4-2：超声波测距仪

用数码管实时的显示测量的距离值

元器件列表：

1. Nduino HD 主控板×1
2. 超声波测距模块×1
3. 数码管模块（计数）×1
4. 3Pin 2510 连接线（白）×1
5. 4Pin 2510 连接线（红）×1

电路连接：



程序编写:

这里需要用超声波测距传感器模块的专用指令。



这里要注意识别超声波测距传感器是 V1.0 版还是 V2.0 版，必须使用对应的读取指令。

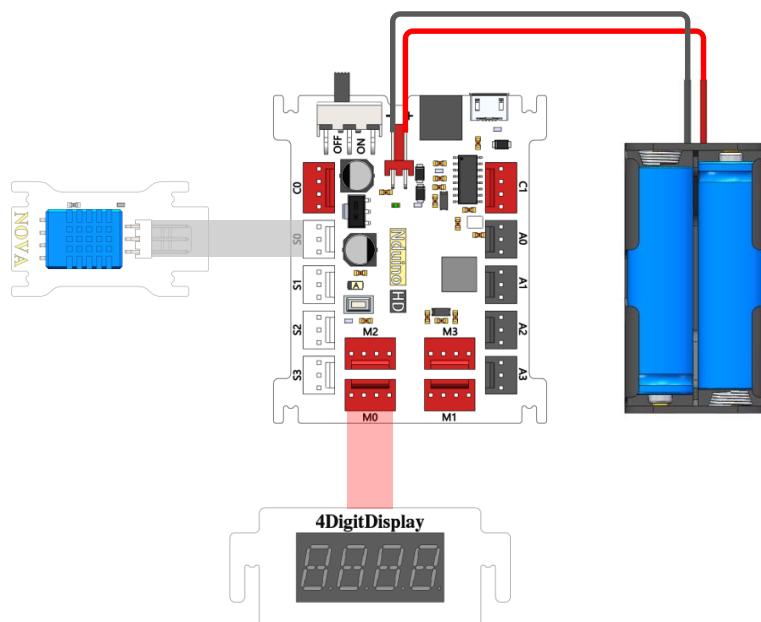
示例 4-3：温湿度计

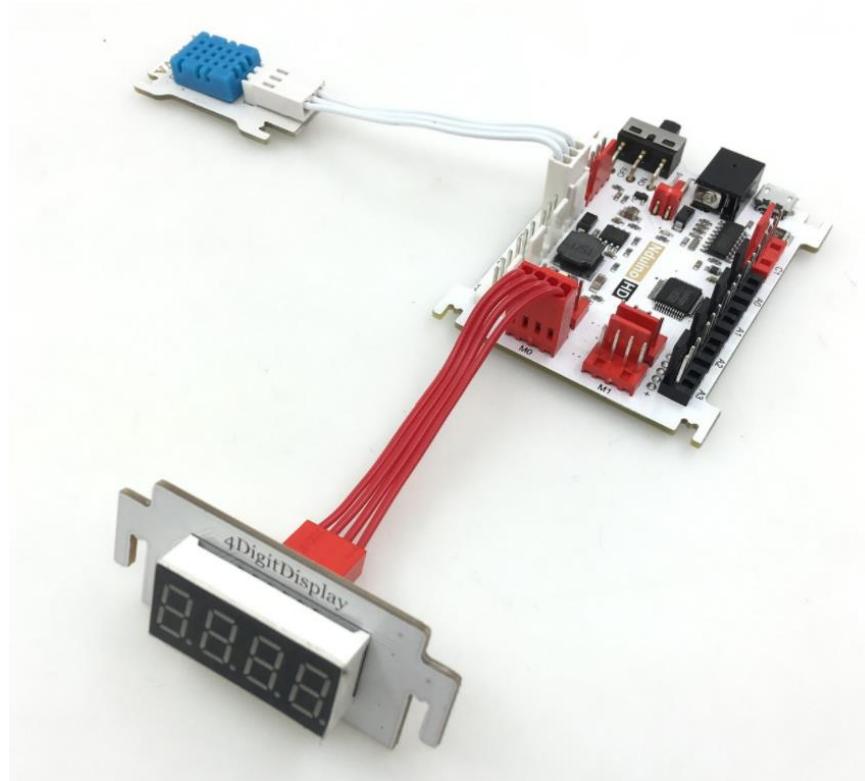
用数码管实时的温度值/湿度值

元器件列表：

1. Nduino HD 主控板 × 1
2. 温湿度传感器 × 1
3. 数码管模块（计数）× 1
4. 3Pin 2510 连接线（白）× 1
5. 4Pin 2510 连接线（红）× 1

电路连接：





程序编写:

这里需要用温湿度传感器的专用指令——读 DHT11。



第五节 模拟量的输出

在第一、二课中同学们已经掌握控制 LED 亮灭的方法，那么 LED 除了开（完全点亮）和关（完全熄灭）两种状态之外，是不是能做到 LED 亮了，但是比完全点亮暗一些呢？

手机、平板可以说是大家使用最多的电子设备，大家会在晚上关灯后调低显示屏的亮度以保护眼睛，在白天强光下调高亮度以便看的更清楚。这些显示屏的背光光源都是 LED，LED 的调光是如何实现的呢？先从 LED 的闪烁说起。

示例 5-1：LED 调光

在编程实现 LED 的调光之前，同学们可以先回顾下示例 2-1：LED 的闪烁。

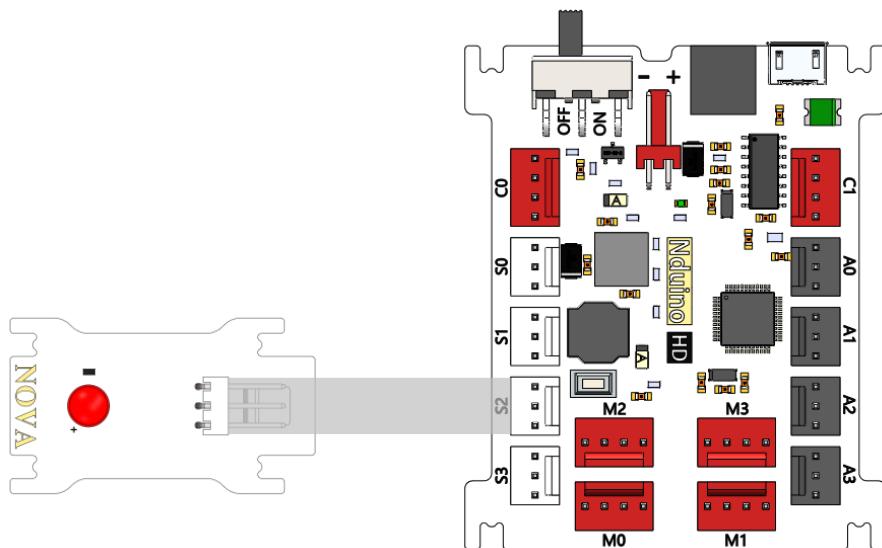
元器件列表：

Nduino HD 主控板 × 1

LED 模块 × 1

3Pin 2510 连接线（白）× 1

电路连接：



LED 灯闪烁的程序已经实现，接下来做几个小实验：

小实验 1：LED 灯亮 100 毫秒，LED 灭 900 毫秒

程序：

好好搭搭硬件程序

重复执行

设置数字口 S2 输出为 0

等待 0.1 秒

设置数字口 S2 输出为 1

等待 0.9 秒

实验 1 直观结果：LED 闪烁，熄灭时间比亮起时间长。

小实验 2：LED 灯亮 10 毫秒，LED 灭 90 毫秒

程序：

好好搭搭硬件程序

重复执行

设置数字口 S2 输出为 0

等待 0.01 秒

设置数字口 S2 输出为 1

等待 0.09 秒

实验 2 直观结果：LED 快速闪烁，注意与实验 3 的直观结果做比较。

小实验 3：LED 灯亮 90 毫秒，LED 灭 10 毫秒

程序：



实验 3 直观结果：LED 快速闪烁，看上去比实验 2 亮多了。

小实验 4：LED 灯亮 90 微秒，LED 灭 10 微秒

因为等待指令的单位为秒，不能更改，如果要让程序暂停的时间是微秒级，需使用等待指令。

程序：



实验 4 直观结果：LED 看上去完全不闪了，注意与实验 5 直观结果做比较。

小实验 5：LED 灯亮 10 微秒，LED 灭 90 微秒

程序：

好好搭搭硬件程序

重复执行

设置数字口 S2 输出为 0

延时 10 微秒

设置数字口 S2 输出为 1

延时 90 微秒

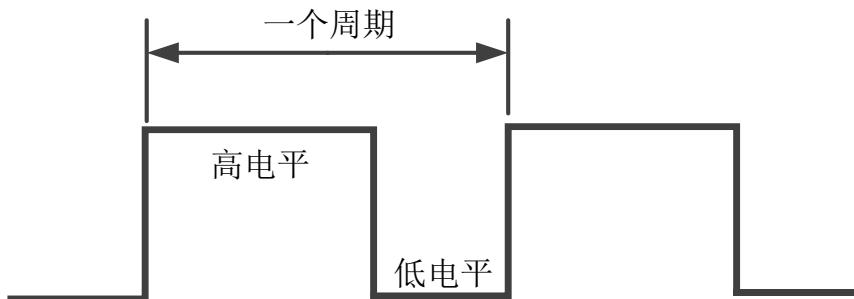
实验 5 直观结果：LED 看上去完全不闪，但是比实验 4 的亮度低很多。

大家可以再试试“0 100”“20 80”、“40 60”、“50 50”、“60 40”、“80 20”、“100 0”不同的组合，延时单位为微秒。可以看到这些参数组合的结果：亮度都不同。

结论：LED 的调光是通过在一定的时间间隔内，调节点亮和熄灭 LED 的相对时间来实现的。

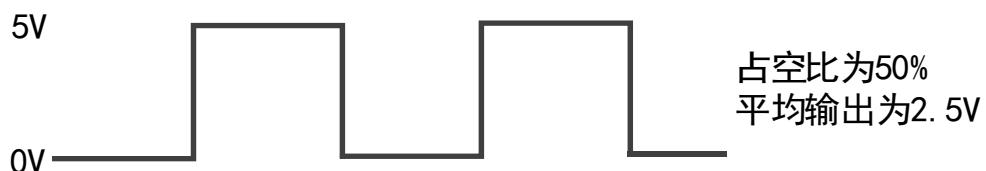
认识新概念——PWM

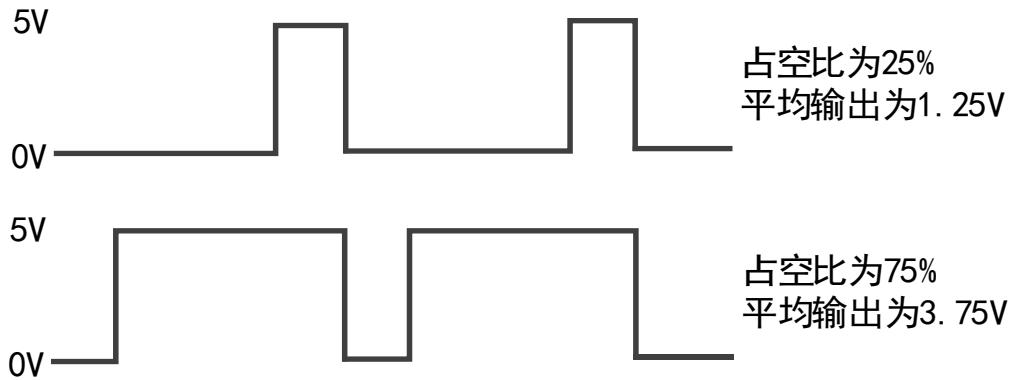
类似 LED 调光这种通过调节供电“通”和“断”的相对时间来改变平均输出大小方法，在工程上称为 PWM（脉冲宽度调制）。



其中，一个周期中高电平持续时间与低电平持续时间的比值，称为占空比。

如高电平电压为 5V，占空比 50% 的 PWM 信号的平均电压为 2.5V；占空比 25% 的 PWM 信号的平均电压为 1.25V；占空比 75% 的 PWM 信号的平均电压为 3.75V。





因为 NOVA 主控器运行程序是单线程的，如果在复杂程序里要调光，运行其他部分程序的时间也会被计算到点亮或熄灭 LED 的延时时间中去，导致亮度调节不准确。所以在复杂程序中需要实现调光功能时，需要调用 **PWM 输出指令**。

认识新指令——设置 PWM 输出指令

设置PWM口 S0 输出为 0

PWM 输出指令能调用主控板的内部定时器，来实现平均输出的调节，无需使用等待或延时指令。NOVA 主控板上能输出 PWM 信号的端口是 S0、S1、S2、S3

PWM 输出指令的数值范围为 0 到 255，输出数值为 0 时，输出电压为 0V，即一直为低电平，相当于数字口输出设置为 0；输出数值为 255 时，输出电压为 5V，即一直为高电平，相当于数字口输出设置为 1。

利用 PWM 输出指令来设置 LED 的亮度的程序为：

好好搭搭硬件程序
重复执行
设置PWM口 S2 输出为 150

同学们试试不同的输出值，看看 LED 的亮度变化。

示例 5-2：旋钮调光

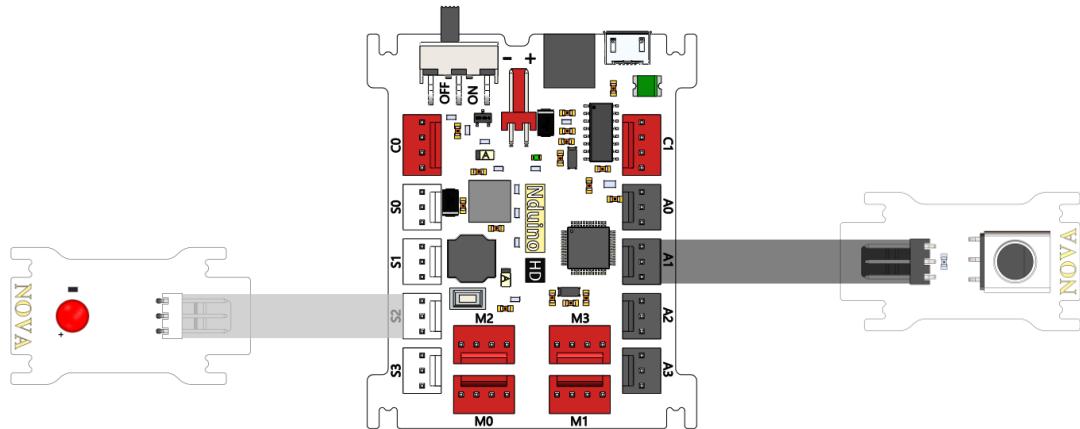
通过前面的内容，了解了模拟量输入和模拟量输出都是范围取值，那么如何用模拟量输入控制模拟量输出呢？

元器件列表：

1. Nduino HD 主控板 ×1
2. 电位器模块 ×1

3. LED 模块x1
4. 3Pin 2510 连接线（白）x1
5. 3Pin 2510 连接线（黑）x1

电路连接:



程序编写:

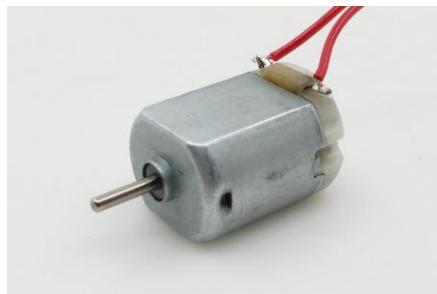


其中电位器的取值范围为 0 到 4095，而 PWM 输出的取值范围为 0 到 255，所以从电位器模块读取到的模拟值需要经过换算，让其最大值不超过 255，最小值不小于 0。

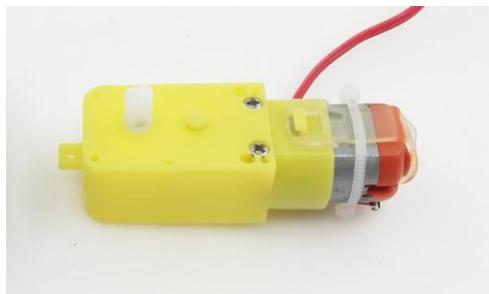
第六节 电机的驱动

电动机是将电能转换为机械能的一种执行器，通常简称为电机。让机构旋转起来，最简单直接的方法就是使用电机。

认识新模块——电机



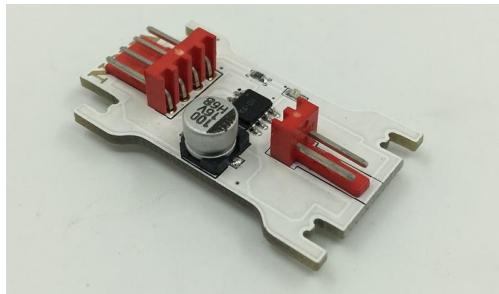
130 电机（直流电机）



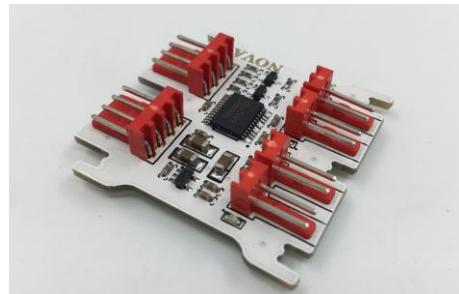
TT 电机（直流减速电机）

认识新模块——电机驱动模块

电机的驱动电压和电流普遍较大，直接接在 NOVA 主控板上是无法正常驱动电机的，所以需要使用电机驱动模块。



单电机驱动模块 (HD-3A)



双电机驱动模块

认识新指令——HD-3A 电机输出指令

HD-3A 电机 M0 输出为 150

认识新指令——双电机输出指令

双电机驱动 M0 电机输出 150

以上两种电机输出指令的数值输出范围均为-255 到 255。能为电机提供驱动信号的端口为 M0、M1、M2、M3。

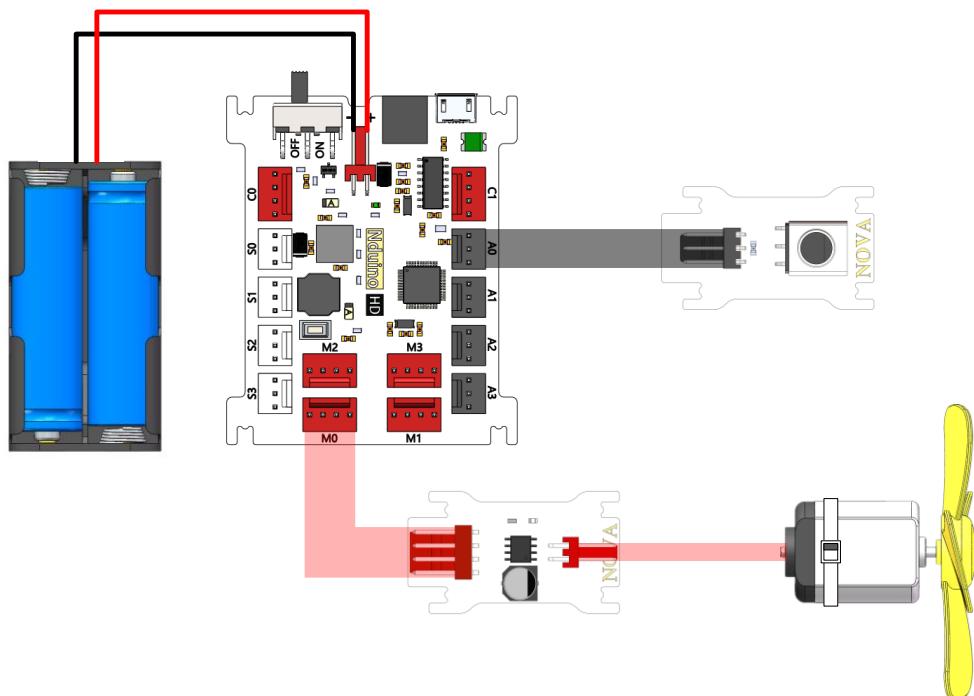
示例 6-1：可调速电风扇

元器件列表：

1. Nduino HD 主控板 × 1

2. 电位器模块 ×1
3. 单电机驱动模块 ×1
4. 130 电机 ×1
5. 风扇叶片 ×1
6. 4Pin 2510 连接线（红）×1
7. 3Pin 2510 连接线（黑）×1

电路连接：



程序编写

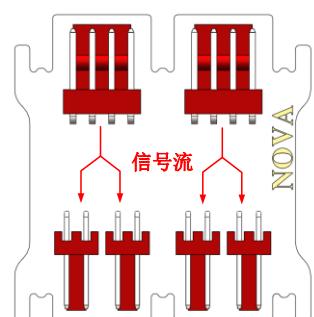


示例 6-2：双电机驱动

使用上相当于单电机驱动模块集成在一起。

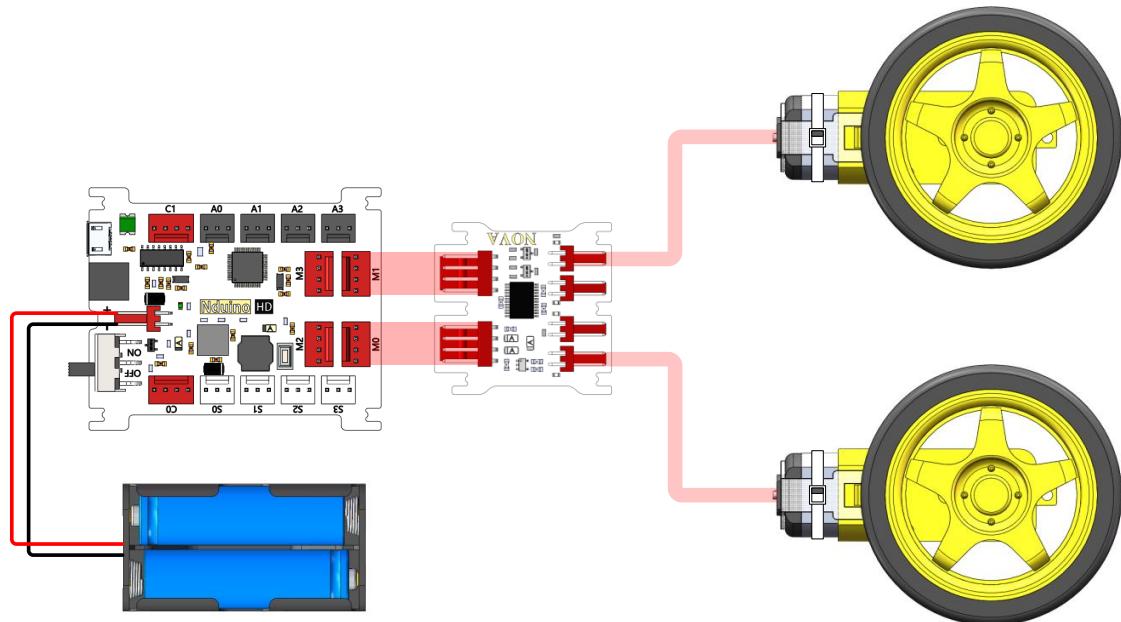
元器件列表：

1. Nduino HD 主控板 ×1
2. 双电机驱动模块 ×1
3. 4Pin 2510 连接线（红）×2

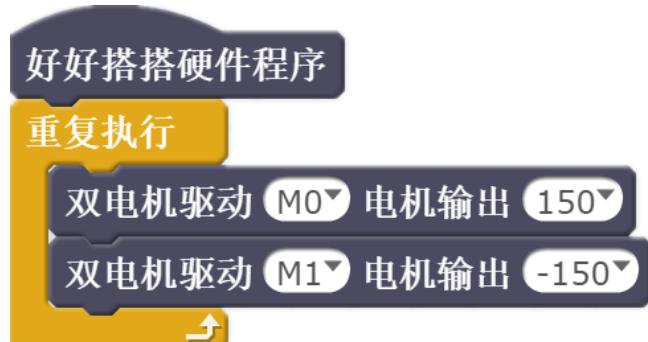


4. 130 电机/TT 电机 × 2

电路连接：



程序编写



试修改电机 1 和电机 2 的转速数值。

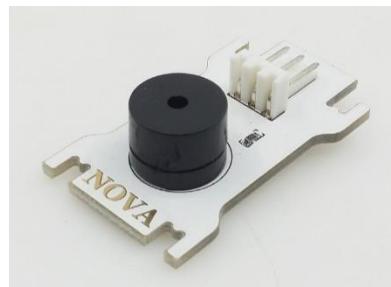
第七节 蜂鸣器

蜂鸣器模块用于发出一定频率的电子声。我们用它编制一首乐曲吧！

声音的三个主观属性分别是音量（响度）、音调和音色。音量指人耳感受到的声音强弱；音调指人的听觉能分辨一个声音的调子高低的程度；音色指声音的感觉特性，即根据不同的音色，即使在同一音高和同一声音强度的情况下，也能区分出是不同乐器或人发出的。

蜂鸣器的驱动电流（PWM 占空比）决定了蜂鸣器发声的音量；蜂鸣器的工作频率（PWM 频率）决定了蜂鸣器发声的音调；蜂鸣器的内部结构与发声原理决定了蜂鸣器发声的音色。

认识新模块——蜂鸣器



认识新指令——设置 PWM 频率指令

设置 PWM S0 频率为 523

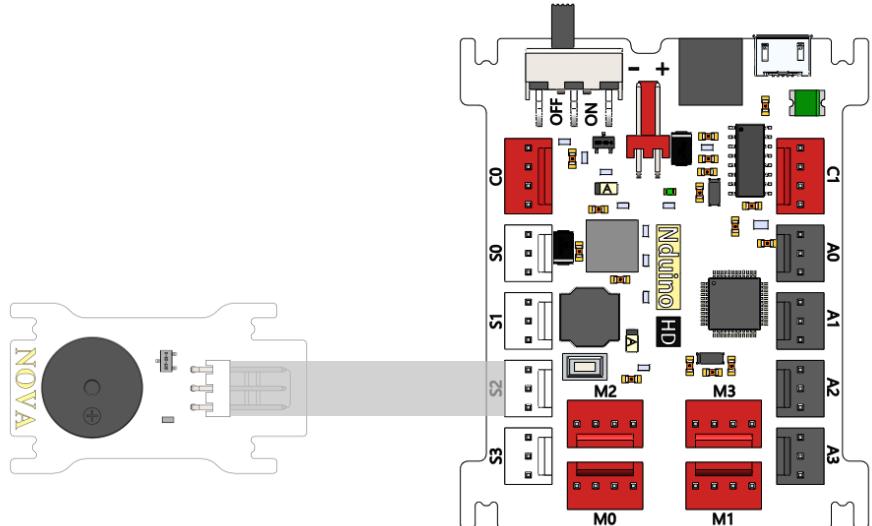
设置 PWM 频率指令是用于修改 PWM 周期的指令，从而实现修改蜂鸣器的音调。

示例 7-1：两只老虎

元器件列表：

1. Nduino HD 主控板 ×1
2. 蜂鸣器模块 ×1
3. 3Pin 2510 连接线（白） ×1

电路连接：



程序编写：

蜂鸣器鸣响程序范式：



发出 C 调音符

我们听到的音乐，每个音符都有固定的频率，比如 C 调音符相对应的频率如下图所示：

| | | | | | | | |
|-------|------|------|------|------|------|------|------|
| C 调音符 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 频率 | 262 | 293 | 329 | 349 | 392 | 440 | 494 |
| C 调音符 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 频率 | 523 | 586 | 658 | 697 | 783 | 879 | 987 |
| C 调音符 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 频率 | 1045 | 1171 | 1316 | 1393 | 1563 | 1755 | 1971 |

《两只老虎》的简谱如下：

两 只 老 虎

1 = bE **$\frac{4}{4}$**

中速

1 2 3 1 | 1 2 3 1 | 3 4 5 - | 3 4 5 - |

两 只 老 虎， 两 只 老 虎， 跑 得 快， 跑 得 快，

5 6 5 4 3 1 | 5 6 5 4 3 1 | 1 5 1 - | 1 5 1 - ||

一 只 没 有 耳 朵， 一 只 没 有 尾 巴， 真 奇 怪， 真 奇 怪。

要播放一首乐曲，除了要发出确定的音调，还需要有节拍的配合。如上图中



两拍的时间长度是一拍的 2 倍，半拍的时间长度是一拍的一半。

一拍的时间长度没有固定的限制，可以是 0.5 秒也可以是 2 秒，时间越短节奏越快。

那么在程序中通过设置蜂鸣器模块的“持续时间”来实现某个音的拍数

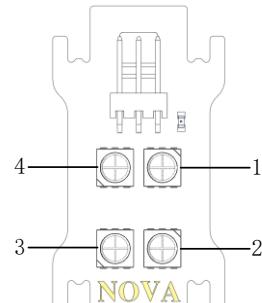
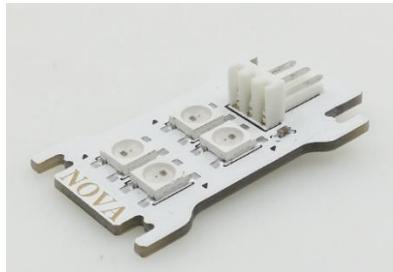
接下来开始编写曲子吧！

第八节 RGB 七彩灯

前面课程用到的 LED，只能发出固定颜色的灯光，当需要发出更多颜色的光时，可以使用 RGB 模块。很炫哦！

RGB 色彩模式是工业界的一种颜色标准，是通过对红(R)、绿(G)、蓝(B)三个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色的，RGB 即是代表红、绿、蓝三个通道的颜色，这个标准几乎包括了人类视力所能感知的所有颜色，是目前运用最广的颜色系统之一。

认识新模块——RGB 模块



认识新指令——RGB 复位指令



RGB 复位指令是点亮 RGB 模块所必需的开始指令，即要点亮或修改 RGB 模块的颜色，必需先使用 RGB 复位指令。

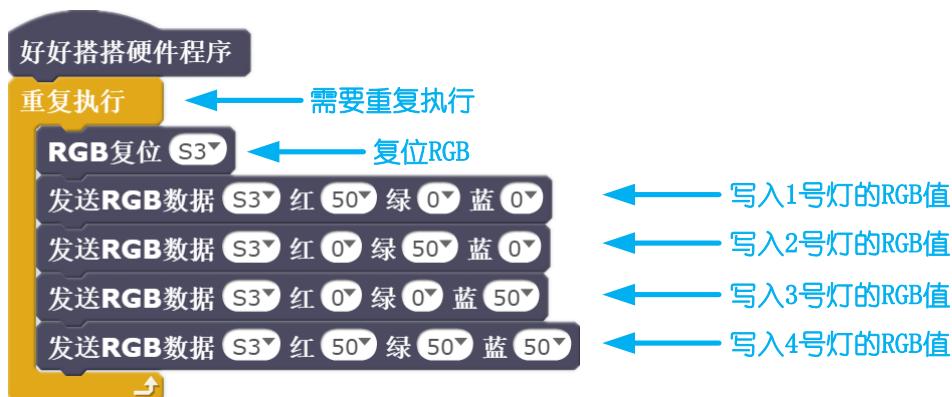
认识新指令——发送 RGB 数据指令



发送 RGB 数据指令是设置 RGB 灯的颜色的指令，发送的第 1 条指令是用于设置第 1 个 RGB 灯颜色的指令，第 2 条 是用于设置第 2 个 RGB 灯颜色的指令，以此类推，发送的第 N 条指令是用于设置第 N 个 RGB 灯颜色的指令。

所以 RGB 模块上有 4 个 RGB 灯，则需要 4 条发送 RGB 数据指令；灯带则是需要点亮多少个 RGB 灯，就需要多少条发送 RGB 数据指令。

RGB 模块点亮程序范式 1：



RGB 模块点亮程序范式 2:



这里需要注意的是 RGB 复位指令和多条发送 RGB 数据指令必须连在一起，中间不能插入任何其它指令。

范式 1 的优点是可以灵活的设置各个 RGB 灯的颜色，缺点是程序体积大，不适合用于设置灯带。

范式 2 的优点是可以快速的设置多个 RGB 灯的颜色，缺点是不够灵活，颜色分布是规则的，适合用于设置灯带。

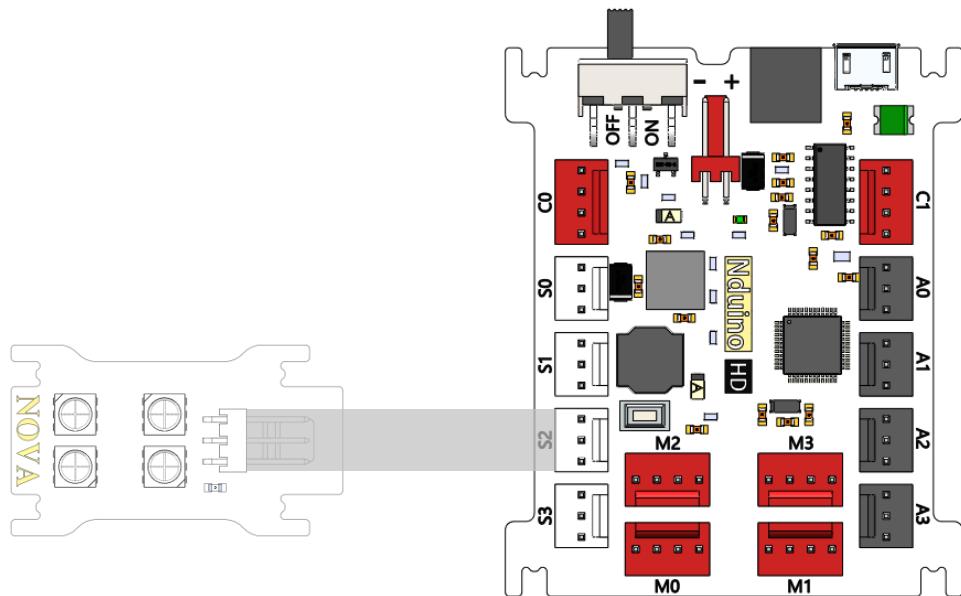
示例 8-1：RGB 交通灯

用 RGB 模块实现交通灯的红黄绿灯变换。

元器件列表：

1. Nduino HD 主控板 ×1
2. RGB 模块 ×1
3. 3Pin 2510 连接线（白） ×1

电路连接：



程序编写：



第二部分

编程入门

第九节 编程基础知识介绍

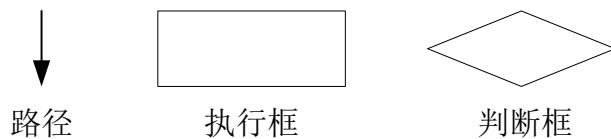
程序是一套人与计算机沟通的语言，既然是语言就有它一套独特的表达方式。

学习编程很像学习一门外语。然而，程序语言比任何一种人类语言都要古板，任何规范之外的表达方式都会被认为是错误的。这套规则来自一代又一代程序设计师的设定，随着时间的流逝，渐渐成为了计算机的行业规范。不了解程序语言的规范，是写不出正确程序的。

程序流程图

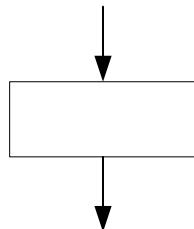
无论是代码程序还是图形化程序，当程序复杂到一定程度之后，都会变得难以读懂。工程上会利用各种图形来表达程序的结构和运行顺序，其中程序流程图是最简单最流行的一种。

程序流程图中最主要的三种符号：

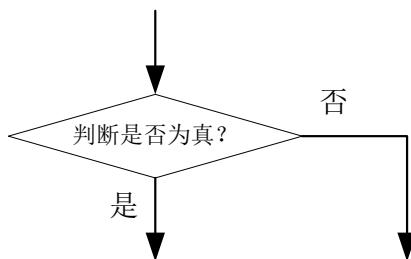


路径表示程序之间连接与流转关系，路径互相之间不能交叉。

执行框代表程序中的一个处理或者步骤，它只能接一条流入路径和一条流出路径，如：



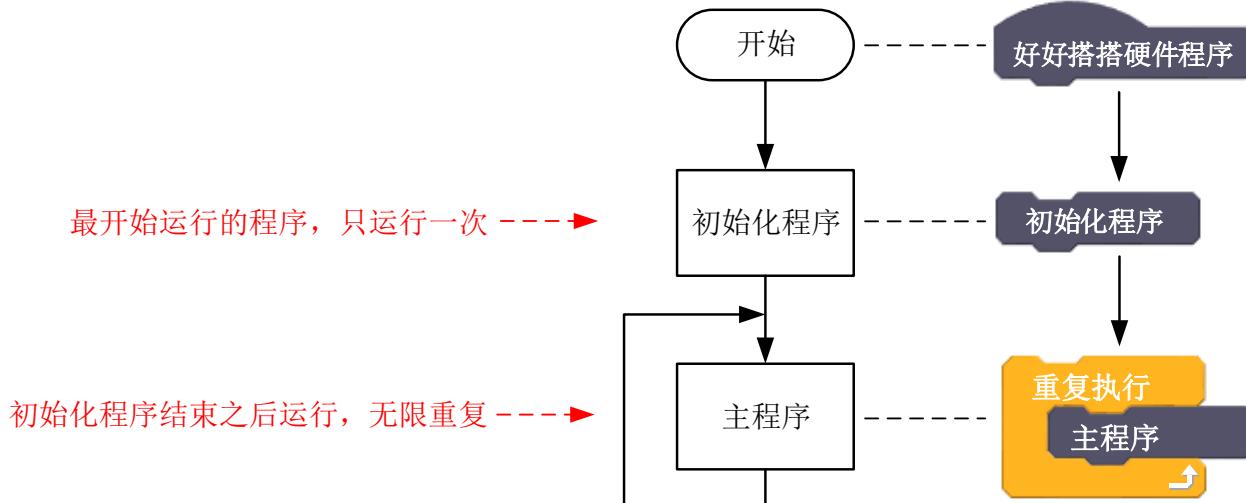
判断框是对一个条件进行判断抉择，它只能有一条流入路径，如果为“真”走一条流出路径，如果为“假”则走另一条流出路径，不可能同时往两条流出路径走，如：



Arduino 开源硬件的程序框架

Arduino 是目前全球最流行的开源硬件，其设计初衷是让不具备专业电子电路知识的创意设计师使用的，所以简单易上手成为了其最大的特点。NOVA 系列开源硬件也是 Arduino 大家族中的一员，其程序框架是相同的。

Arduino 开源硬件的程序框架包括两部分：初始化程序和主程序。程序运行的流程如下：



好好搭搭硬件程序是程序的开始指令，有且只允许有一个，其它指令必须连接在其之后，所有与其断开的指令都不会运行。

初始化程序在程序中的作用一般是用于设定硬件启动时初始状态的，位于“好好搭搭硬件”之后，“主程序”之前，在程序开始后率先运行一次，之后不再运行。例如示例三交通灯程序中，初始化程序的作用就是使接通电源时，三个 LED 都处于熄灭状态。

主程序是一直重复运行的，位于“重复执行”指令之中。重复执行指令在程序中有且只允许有一个。

第十节 数字量输入与条件判断指令

数字量传感器

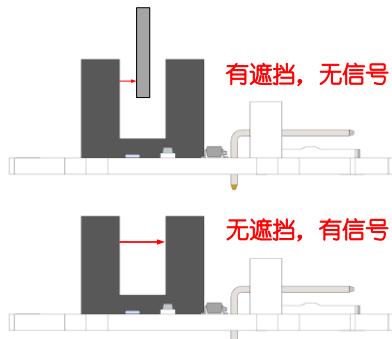
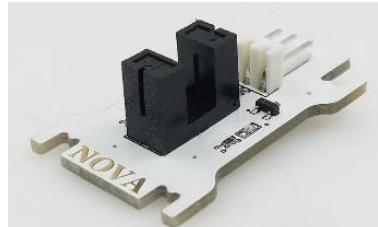
数字量是只有“1”和“0”两种状态的量，比如：“是”与“否”、“开”与“关”、“真”与“假”，这些都是非此即彼的，都是数字量，也被称为开关量，只不过计算机系统只能处理数字信息，所以用“1”和“0”来指代。

认识数字量传感器

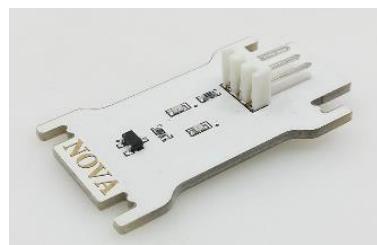
按钮开关：感知外界时候有否物体按压它。



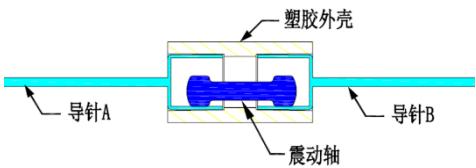
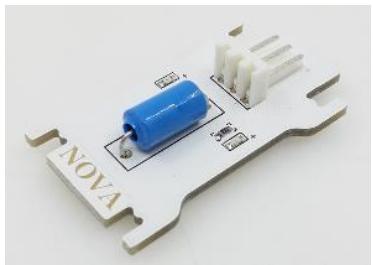
光电开关：槽式光电开关通常是标准的U字型结构，其发射器和接收器分别位于U型槽的两边，并形成一光通路，当被检测物体经过U型槽且阻断光通路时，光电开关的开关量信号产生变化。



霍尔开关：霍尔传感器是根据霍尔效应制作的一种磁场传感器。用于检测附近是否有磁性物体靠近。



震动开关：没有振动时，震动轴呈静止状态，导针A与导针B两端则为接通状态，当有震动时，震动轴会运动，导针A与导针B之间会有瞬间的断开，实现震动触发的作用。

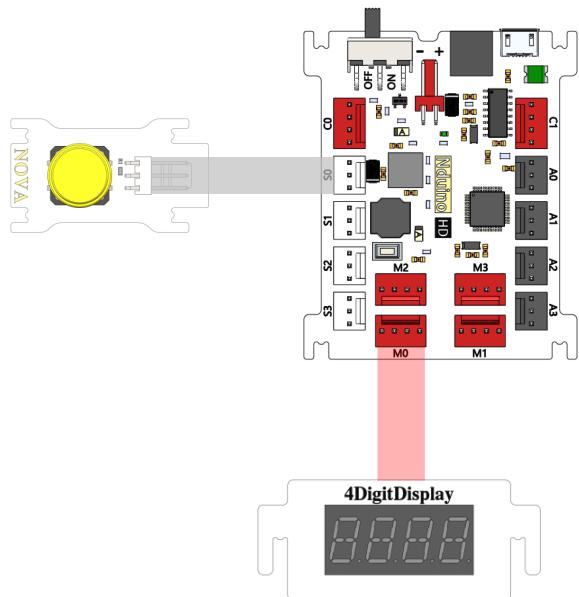


示例 10-1：用数码管显示开关量传感器的状态，以按钮开关为例。

元器件列表：

1. Nduino HD 主控板 ×1
2. 按钮开关模块 ×1
3. 数码管模块 ×1
4. 3Pin 2510 连接线（白）×1
5. 4Pin 2510 连接线（红）×1

电路连接：



程序编写：



上传程序后，可以看到当按钮按下时，数码管显示 1，当按钮松开时，数码管显示 0。大家可以把按钮开关换成其他开关量传感器模块，试试看。

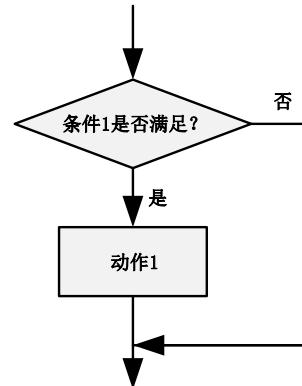
条件判断指令

数字量传感器的使用，通常都要配合条件判断指令来使用，即需要判断数字量传感器输入的数值是“1”还是“0”：如果是“1”，则认为是“真”；如果是“0”，则认为是“假”。

认识新指令——如果-那么指令



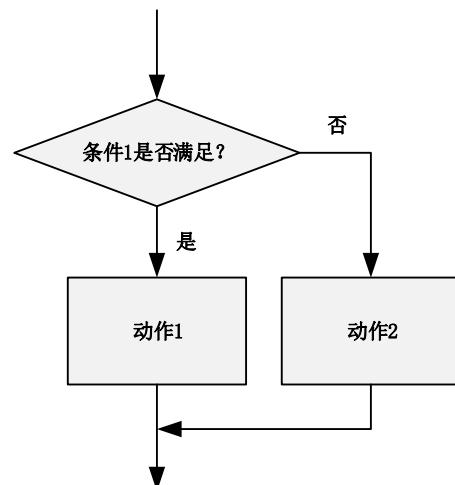
功能说明：如果条件 1 成立，则执行动作 1，否则不执行动作 1



认识新指令——如果-那么-否则指令



功能说明：如果条件 1 成立，则执行动作 1 不执行动作 2，否则不执行动作 1 执行动作 2。

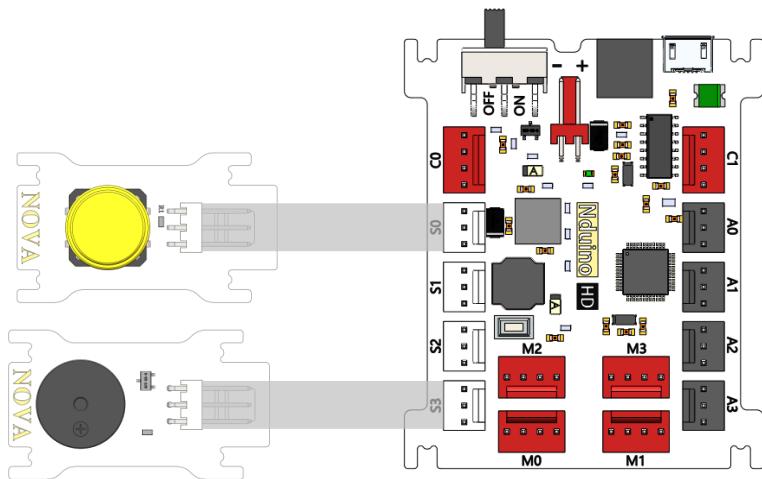


示例 10-2：门铃

元器件列表：

1. Nduino HD 主控板 ×1
2. 按键模块 ×1
3. 蜂鸣器模块 ×1
4. 3Pin 2510 连接线（白）×2

电路连接：



程序编写：



如果-那么指令



如果-那么-否则指令

条件判断指令是硬件编程中最重要的指令。能否灵活巧妙的使用它，可以直接的表现出一个程序员的逻辑思维能力。

后续的课程会不断的使用到条件判断指令，会有更多精妙的应用！

第十一节 变量

概念：

变量可以保存程序运行时用户输入的数据和特定的运算结果。

在程序中使用变量几个主要作用是：

- (1) 提升程序可读性；
- (2) 方便批量修改程序数据；
- (3) 存放程序中间运算结果。

变量的运用是编程最重要的能力之一，能否灵活恰当的使用变量能直接体现对编程理解的深度。

新建变量：



新建完成后，所有已建的变量，都会出现在“数据”脚本类中。



注意，变量名不能含有中文！

认识新指令——赋值指令

用于将用户要输入的数据或中间运算结果写入到变量中。

将 **a** 设定为 **0**

认识新指令——增值指令

用于实现变量值的变化，每运行一次，变量值增加一定数值。

将变量 **a** 的值增加 **1**

认识新指令——变量调用指令

当需要处理变量时使用。

a

变量的作用要在运算、逻辑判断、循环、函数等构成复杂程序逻辑的场合才能真正的体现，所以变量的相关案例和解释将在后续的章节中详细说明。

第十二节 数学与逻辑运算

在程序编写中经常需要对变量或数值进行换算或比较，这时可以调用数学与逻辑运算类指令。

认识四则运算指令：



其中“除运算”的运算结果是含有小数位，如：

将 **a** 设定为 **5 / 2**

程序运行结果，变量 a 的值为 2.5

认识比较指令：



比较指令的运行结果为“真”或“假”，即正确的为真，错误的为假。例如：

| | | | |
|-----------------|---|-----------------|---|
| 3 < 5 | 真 | 3 < 1 | 假 |
| 3 = 3 | 真 | 3 = 5 | 假 |
| 3 > 1 | 真 | 3 > 5 | 假 |

比较指令常配合“如果...那么...”指令使用。

认识布尔运算指令：



布尔运算指令是对真值（真或假）进行操作的指令，达到多个条件的协同判断的目的。

| 布尔运算真值表 | | |
|-----------|-----------|-----------|
| 真 且 真 → 真 | 真 且 假 → 假 | 假 且 假 → 假 |
| 真 或 真 → 真 | 真 或 假 → 真 | 假 或 假 → 假 |
| 真 不成立 → 假 | 假 不成立 → 假 | |

示例 12-1：华氏温度计

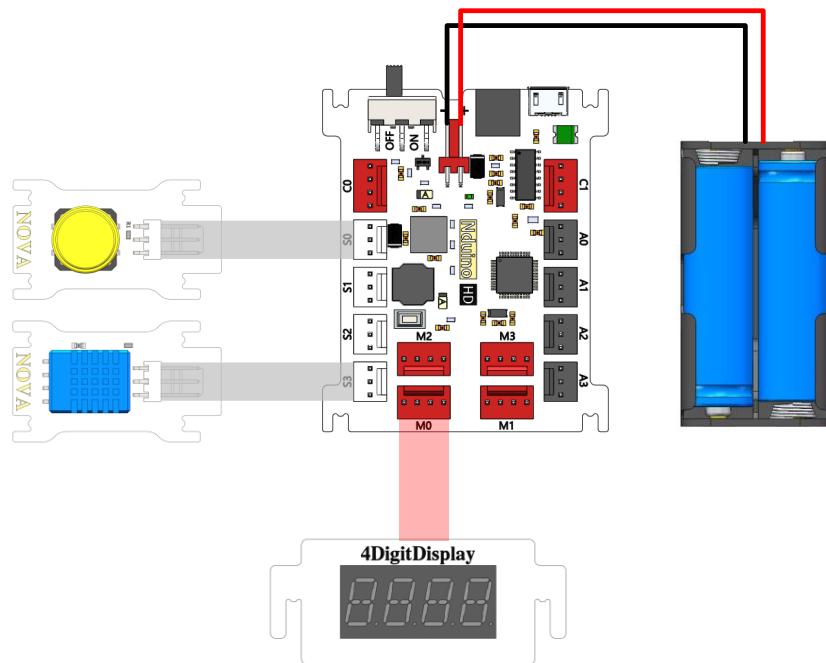
在日常生活中，常见的温度单位又两种：摄氏度和华氏度，前者是我国使用的温度单位 ($^{\circ}\text{C}$)，后者是美国使用的温度单位 ($^{\circ}\text{F}$)，两者的关系是：华氏度=摄氏度 $\times 1.8+32$

通过程序实现：数码管显示摄氏度和华氏度，并用按键做为输入实现切换。

元器件列表：

1. Nduino HD 主控板 × 1
2. 温湿度传感器模块 × 1
3. 数码管模块（数字）× 1
4. 3Pin 2510 连接线（白）× 2
5. 4Pin 2510 连接线（红）× 1

电路连接:



程序编写:



其中，变量 tem_C 和 tem_F 分别指代摄氏度和华氏度。这里变量的作用是提升程序的可读性和存放中间计算结果。

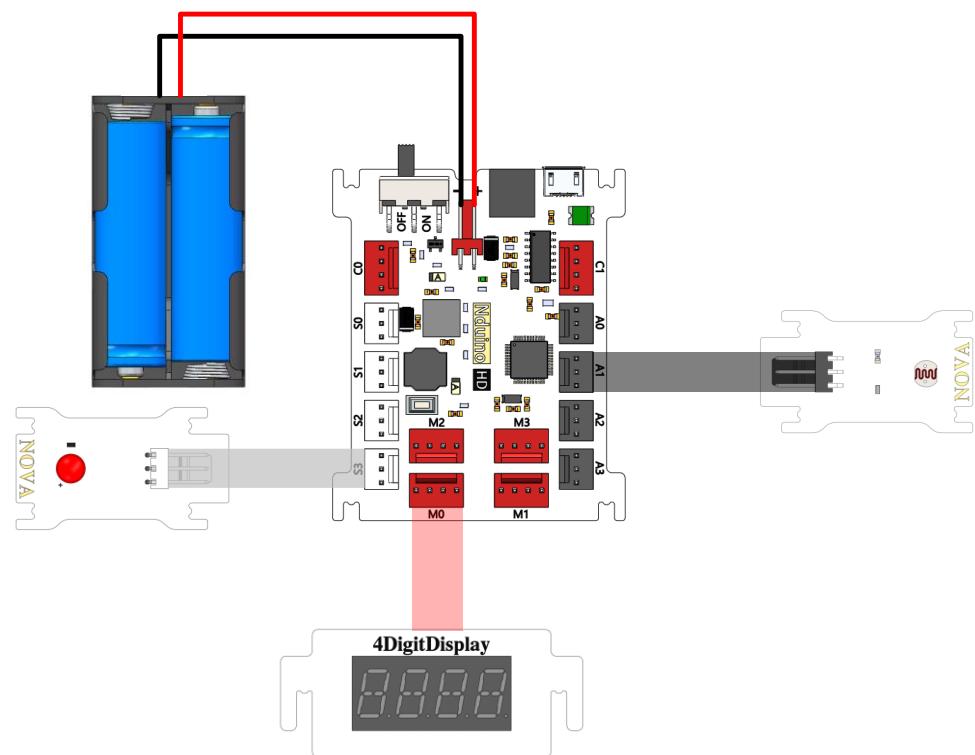
示例 12-2：光控灯（光敏开关版）

白天（光照强度高），LED 自动熄灭；夜晚（光照强度低），LED 亮起，LED 只需开关两种状态。

元器件列表:

1. Nduino HD 主控板 ×1
2. LED 模块 ×1
3. 光敏传感器模块 ×1
4. 数码管模块（计数） ×1
5. 3Pin 2510 连接线（白） ×1
6. 3Pin 2510 连接线（黑） ×1
7. 4Pin 2510 连接线（红） ×1

电路连接：



程序编写：

第 1 步

测量夜晚需要开灯时的光照强度，观察数码管的显示值并记录下来，这个值就是用于比较的阈值。

阈值（Critical Value） 是指两个相邻定义域的边界。比如常说的白天和黑夜的边界，春天和夏天的边界，白色和灰色的边界。有的阈值是约定俗成的，如春天和夏天边界是立夏这一天；有的阈值则是可以自定义的，如冷和热，亮和暗，每个人的标准都不一样。

模拟量输入与阈值进行比较大小，即可将模拟量输入传感器视为一个开关。



第 2 步

根据的第 1 步测量到的阈值，本例中使用的阈值为 50，编写光控开关的程序：



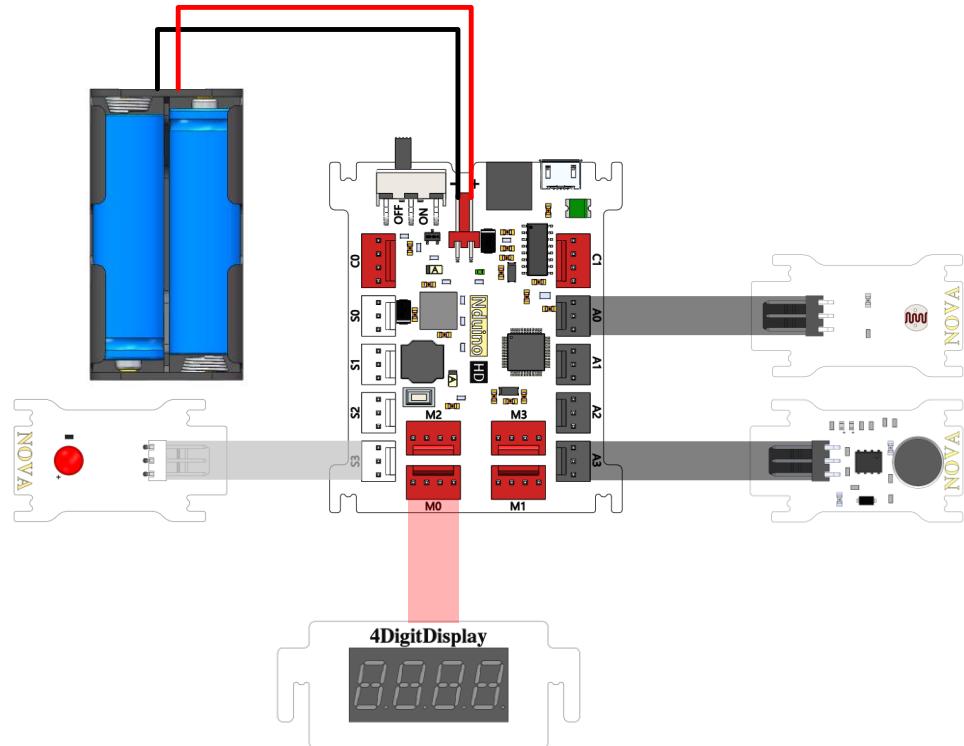
示例 12-3：夜间声控灯

只有在夜间才能通过声音点亮的灯

元器件列表：

1. Nduino HD 主控板 ×1
2. 光敏传感器模块 ×1
3. 声音传感器模块 ×1
4. 3Pin 2510 连接线（白）×1
5. 3Pin 2510 连接线（黑）×2

电路连接:



程序编写:



第十三节 随机数

在编写程序时，尤其是设计游戏和模拟实验时，你有可能需要生成随机数让程序富有变化。

为此，Scratch 专门提供了在…到…间随机选一个数的指令。

认识随机数指令

用于随机从某数字范围中抽取一个数（整数）。

在 1 到 10 间随机选一个数

指令中的两个参数指定随机数的取值范围。随机数指令每运行一次，产生一个随机数，产生的这个数即为该指令的运行结果。试运行以下程序并分析：



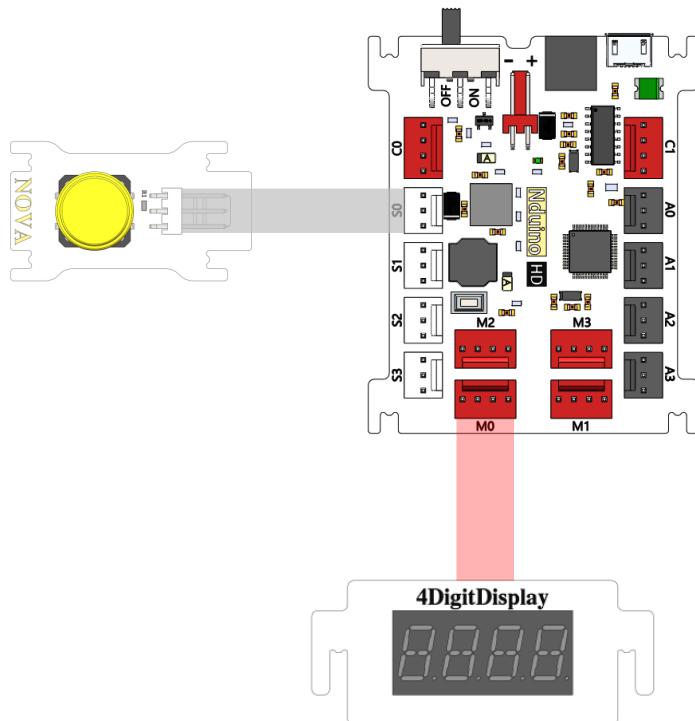
示例 13-1：摇号机（基本版）

从学号尾号 01 到 50 之间抽取一个号码，多次抽取，号码可能出现重复。

元器件列表：

1. Nduino HD 主控板 × 1
2. 按钮模块 × 1
3. 数码管模块（数字）× 1
4. 3Pin 2510 连接线（白）× 1
5. 4Pin 2510 连接线（红）× 1

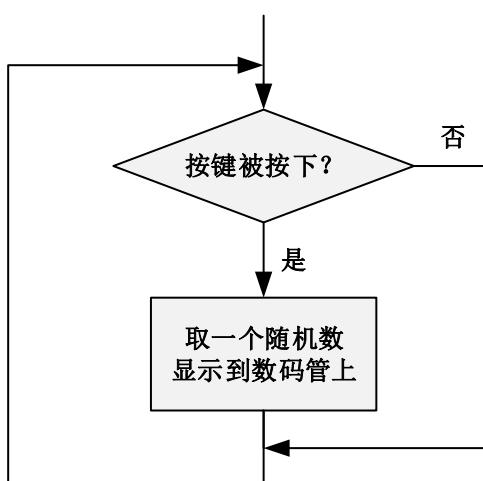
电路连接：



程序编写:



程序说明:



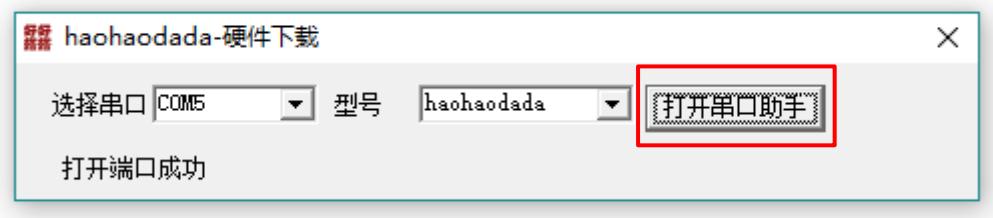
- (1) 当按键被按下，不断运行随机数指令，显示到数码管上，看到不断滚动的数字；
- (2) 当按键被松开，不运行随机数指令，数码管数字不更新，看到一个确定的数字。

第十四节 串口打印

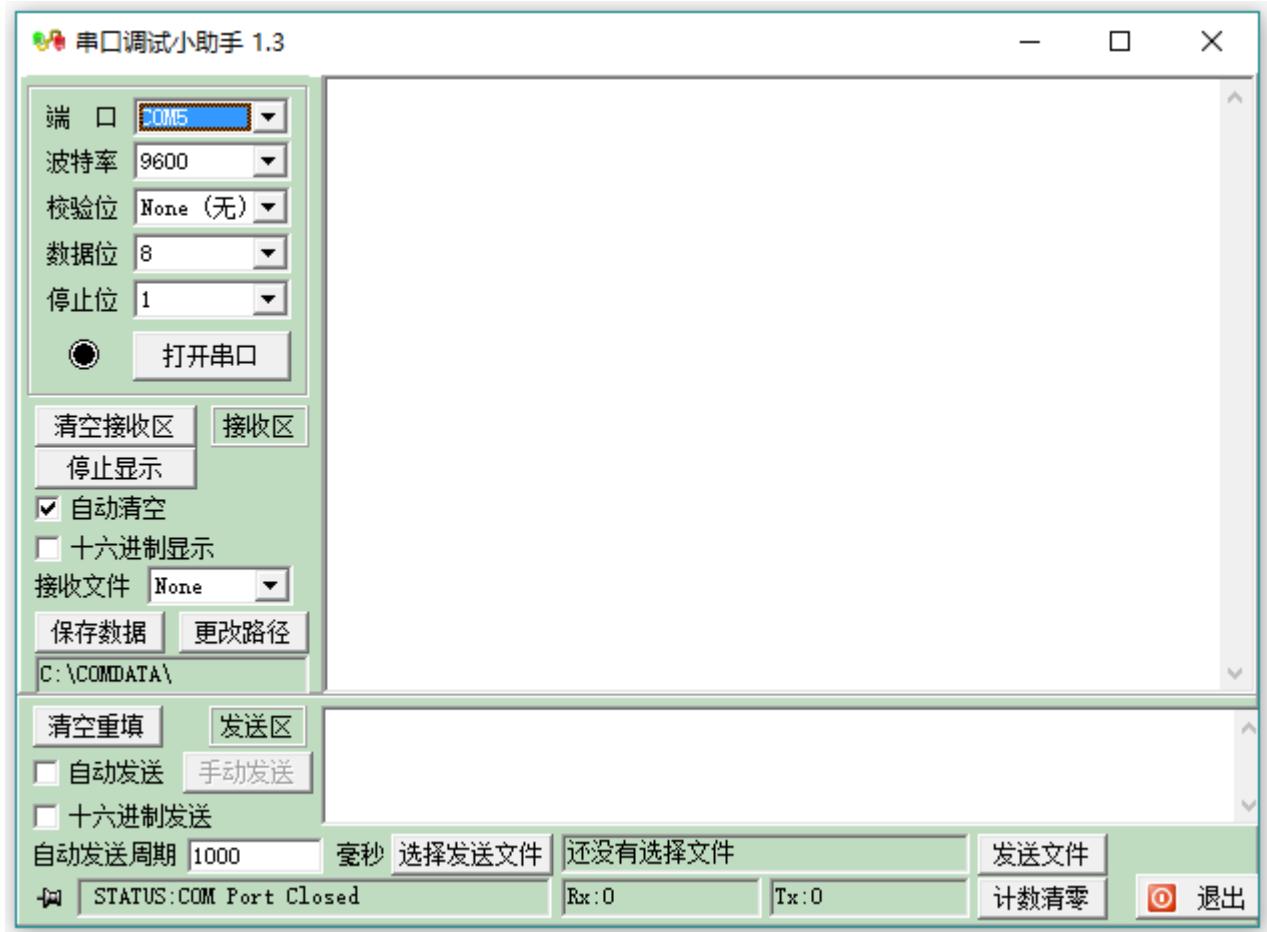
在调试程序时，程序员经常需要查看各种参数或变量的数值，这时可以利用串口助手，将这些信息在电脑屏幕上显示出来。

认识串口助手

在“硬件下载插件”中点击“打开串口助手”按钮。



串口调试小助手界面：



波特率是指通过串口发送数据的速度，即单位时间内发送数据的个数。单位：bps（每秒位数）。

认识串口速率设置指令

串口速率设置 9600

波特率是指通过串口发送数据的速度，即单位时间内发送数据的个数。单位：bps（每秒位数）。

常用的波特率有：1200、2400、4800、9600、19200、38400、57600、115200。NOVA 串口的默认波特率为 9600。

程序中设置的“串口速率”一定要与串口助手中的“波特率”一致，是正常实现串口通讯的必要条件！

认识串口输出指令

串口输出 12345

与数码管显示指令类似，串口输出指令是将数据显示到串口助手中。

重复运行串口输出指令的效果是：



串口输出指令输出的数据，不会自动换行，重复执行之后数据之间会首尾相连，所以串口输出指令常用于连接多个数据打印时使用，如果只输出一个数据，建议使用“串口输出并换行”指令。

认识串口输出并指令

串口输出并换行 12345

串口输出并换行指令也是将数据显示到串口助手中，相比串口输出指令，它会在输出数据结束之后，自动添加一个换行符。

重复运行串口输出并换行指令的效果是：



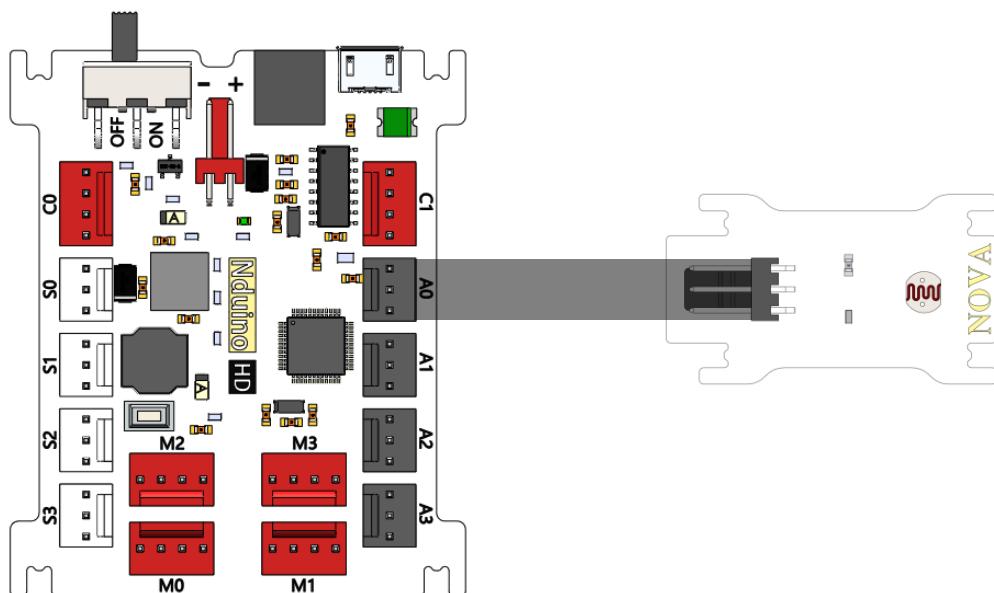
示例 14-1：串口打印单个传感器数值

在串口监视器上显示亮度值。

元器件列表：

1. Nduino HD 主控板 × 1
2. 亮度传感器模块 × 1
3. 3Pin 2510 连接线（黑）×1

电路连接：



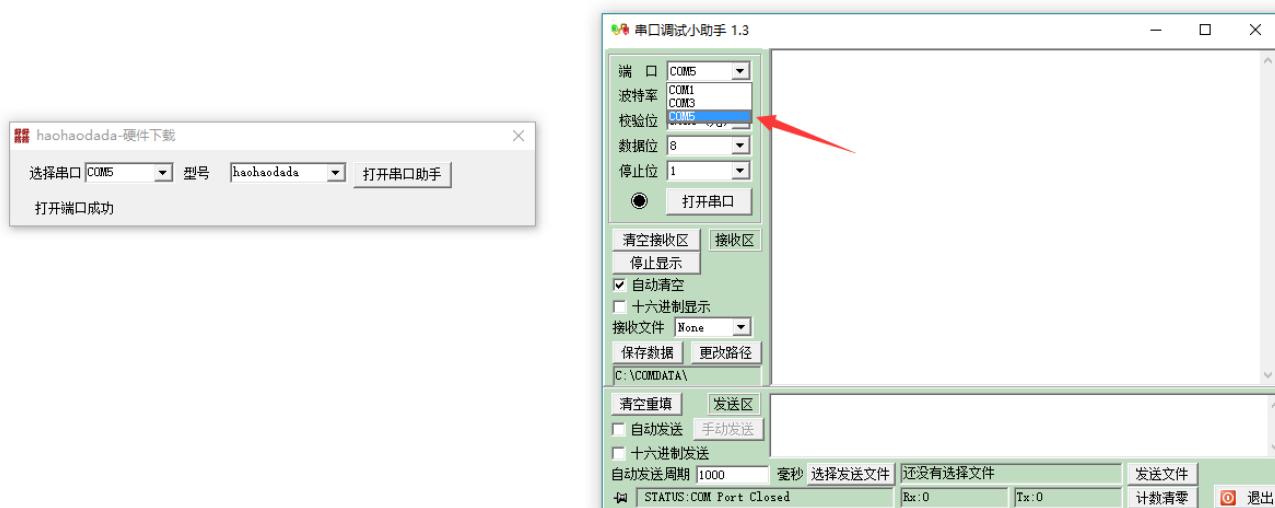
程序编写：



待上传完成之后，打开串口助手。注意！不要断开 USB 连接。

需完成如下三项操作，串口助手方可将数据显示出来。

- (1) 端口选择。端口选择与“硬件下载插件”一致。如本例中硬件下载插件中的串口为 COM5，则串口助手中端口选择也为 COM5；



- (2) 波特率检查。串口助手中的波特率要与好好搭搭程序中的串口速率一致。本例中选用默认的 9600。



(3) 点击“打开串口”按钮，之后，便可在接收区看到数据的显示了。



注意！注意！注意！如果要再次编译下载程序或断开 USB 连接，必须关闭串口！

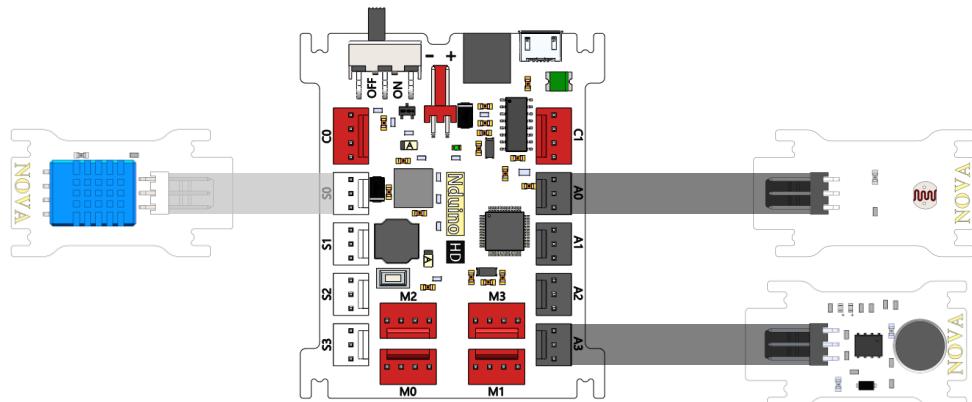
示例 14-2：串口打印多个传感器数值

在串口监视器上显示亮度值、声音值、温度值和湿度值。

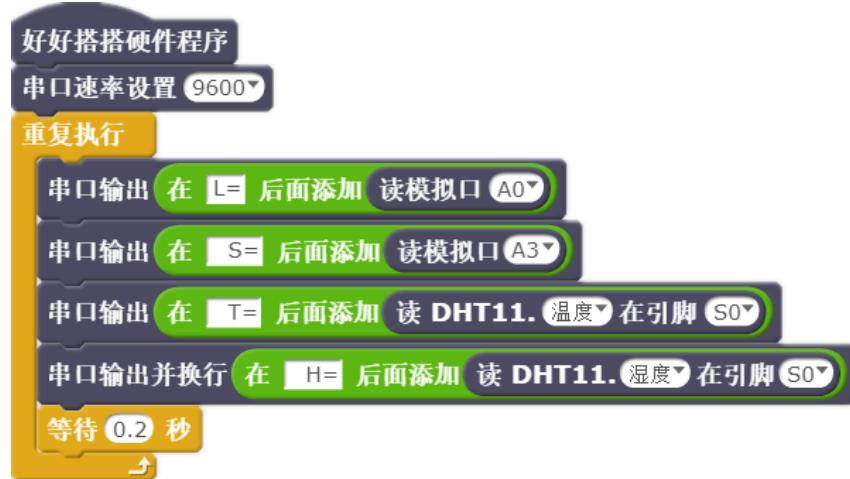
元器件列表：

1. Nduino HD 主控板 × 1
2. 亮度传感器模块 × 1
3. 声音传感器模块 × 1
4. 温湿度传感器模块 × 1
5. 3Pin 2510 连接线（白）× 1
6. 3Pin 2510 连接线（黑）× 2

电路连接：



程序编写：

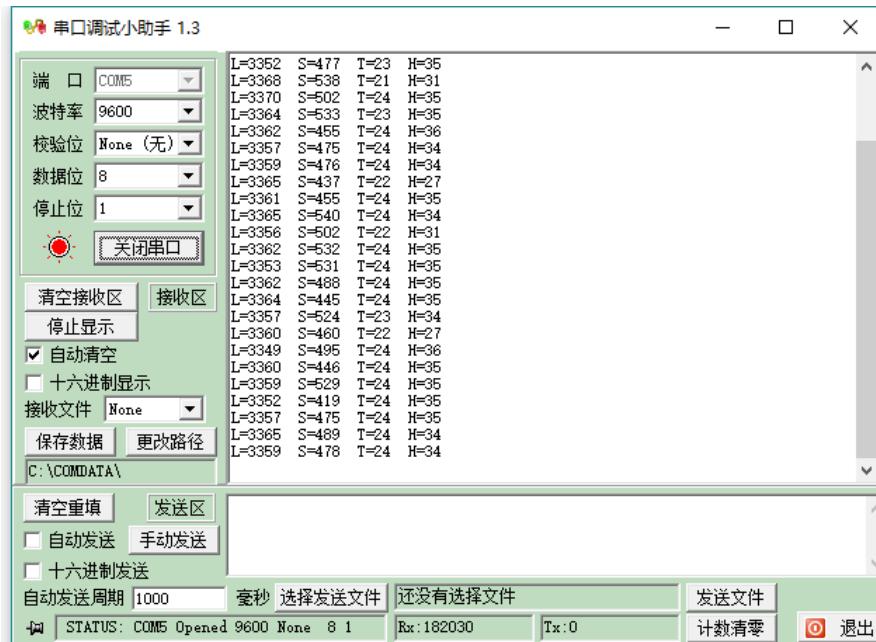


其中，

- (1) L 为 Light 光；S 为 Sound 声音；T 为 Temperature 温度；H 为 Humidity 湿度。
- (2) 为了让所有输出信息，在同一行，所以最后一条输出指令用“串口输出并换行”指令，之前的全部用“串口输出”指令。
- (3) 为让不同相同信息互相隔开，可以用空格符，如：



打印结果为：



第十五节 新建功能块

当程序复杂到一定程度之后，一部分功能相对独立或会被多次使用的程序段通常会采用新建功能块的方式简化程序，提升程序的可读性。

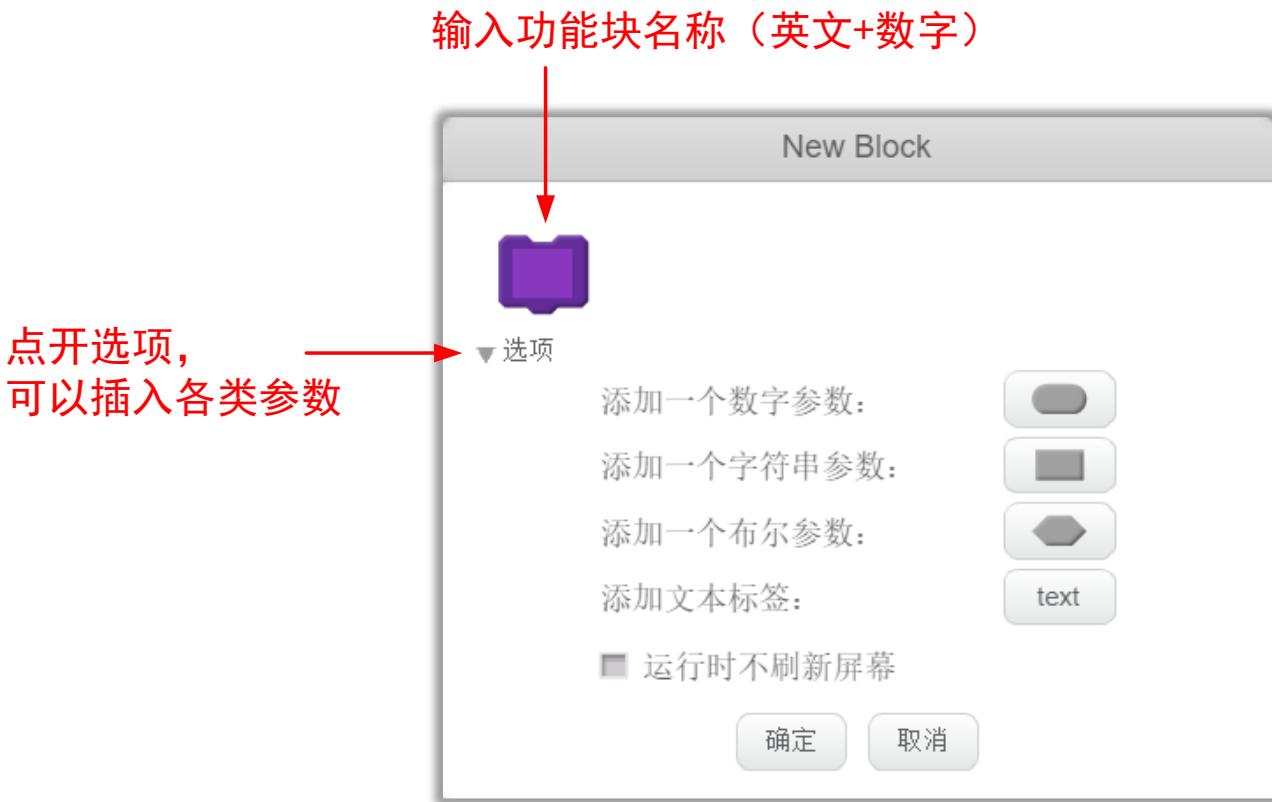
认识新建功能块

在“更多模块”类中有“新建功能块”按钮。



点击“新建功能块”按钮，开始新建功能块：





注意： 功能块名称和参数都不能出现中文！

示例 15-1：RGB 交通灯（功能块版）

在示例 8-1 中，用 RGB 模块实现了交通灯的红黄绿灯变换。其中，RGB 模块的相关指令在使用时有固定的范式：



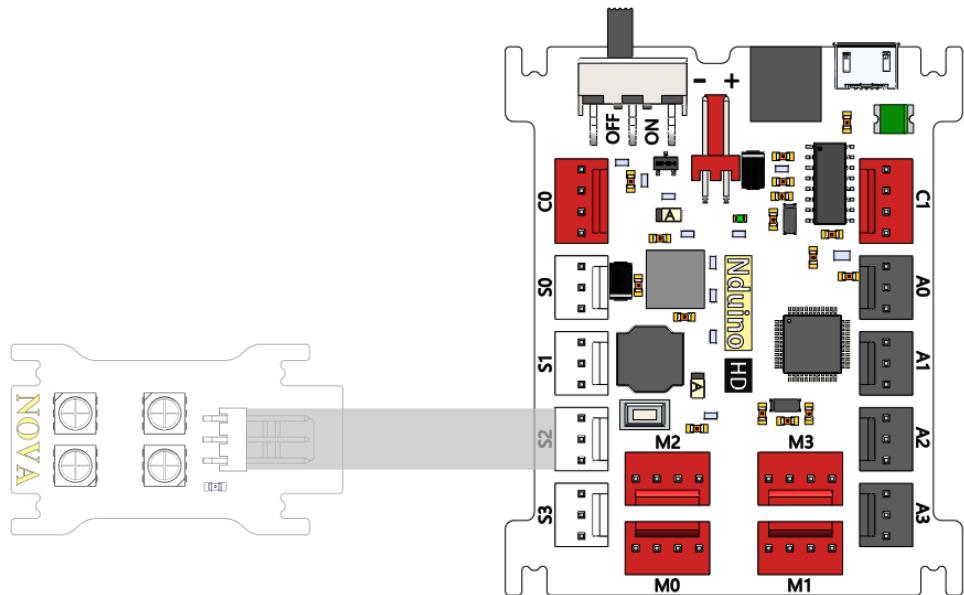
像这样的程序段，就可以采功能块的方式提升程序的可读性。

元器件列表：

1. Nduino HD 主控板 ×1
2. RGB 模块 ×1

3. 3Pin 2510 连接线（白） ×1

电路连接：

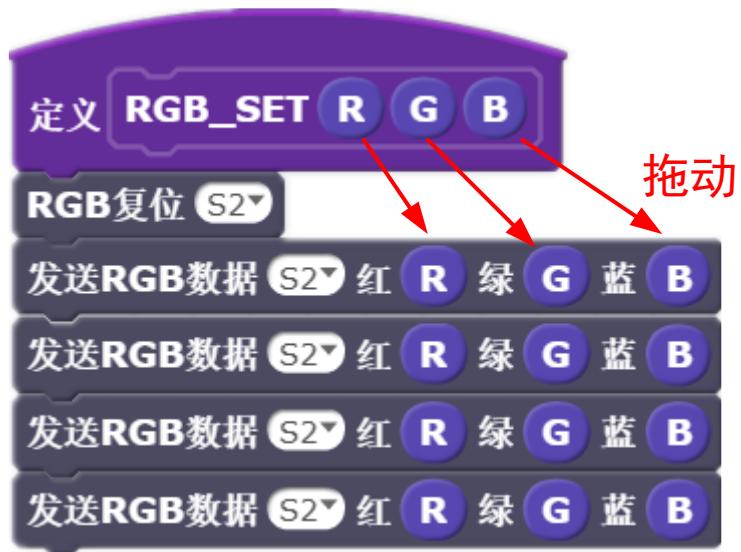


程序编写：

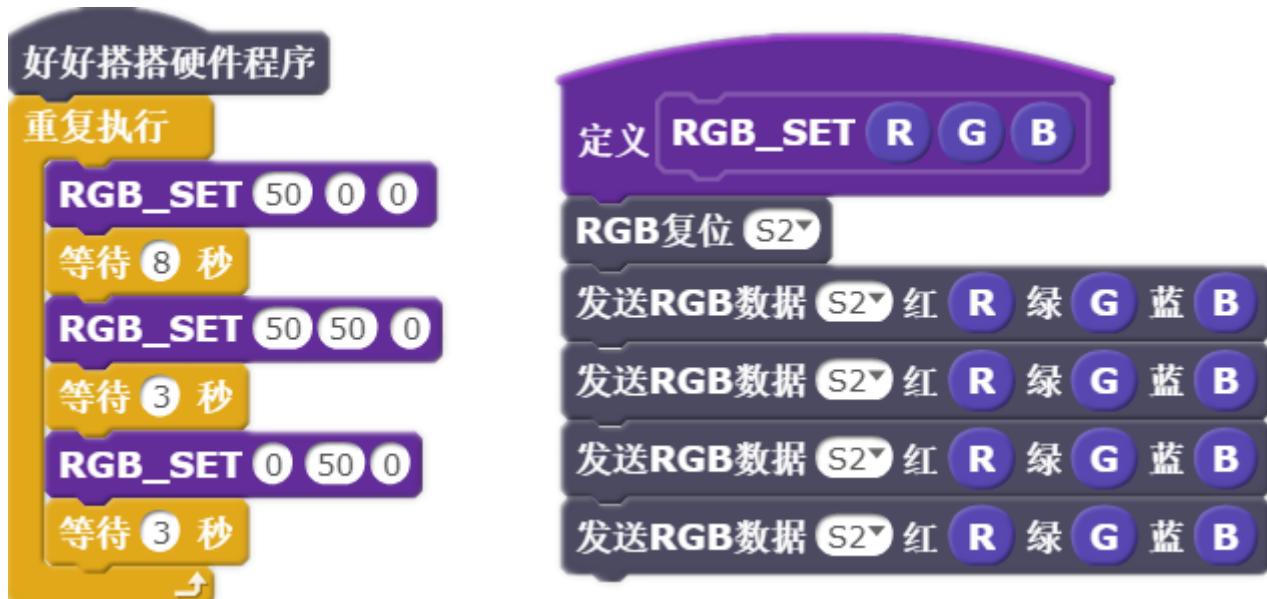
新建带参数的功能块：



在功能块的指令中添加功能块的参数：



程序为：



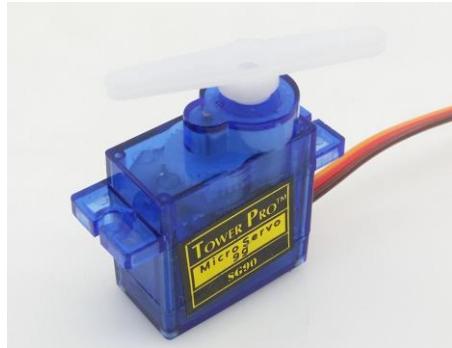
第三部分

硬件进阶

第十六节 舵机控制

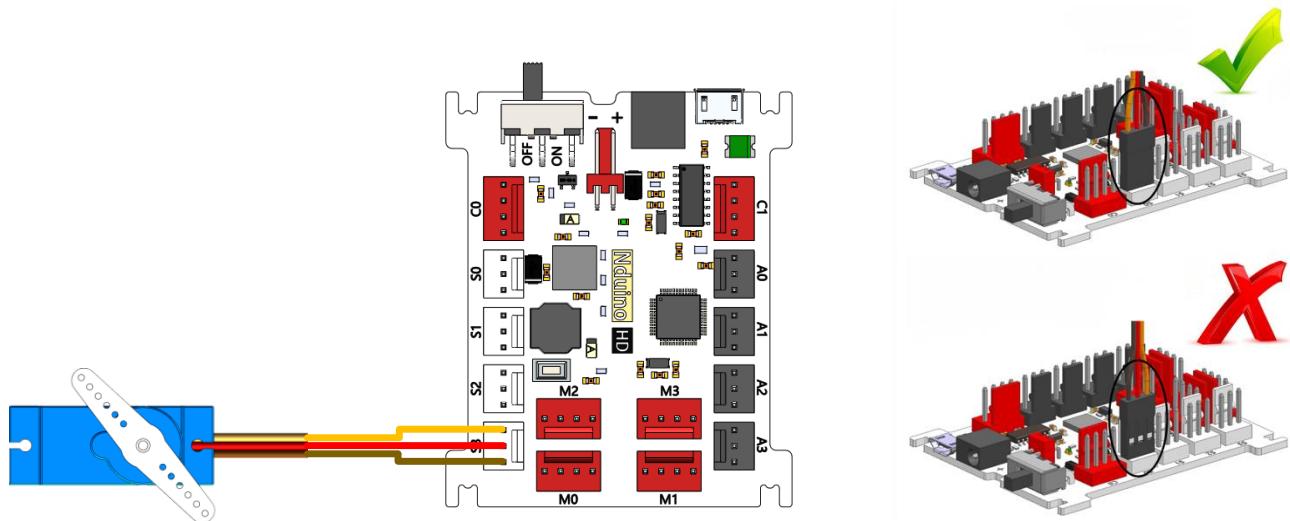
舵机是带有位置反馈的直流电机，是一种伺服电机，能精确控制电机输出轴的转动角度。

认识舵机

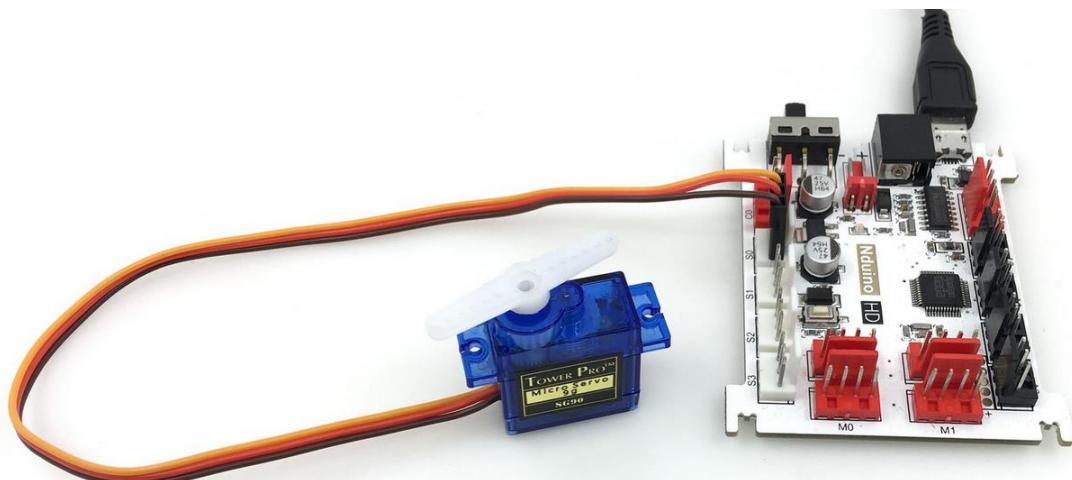


SG90 舵机（转动范围 $0^\circ \sim 180^\circ$ ）

电路连接：舵机可以连接到主控板的白色或黑色接口。



舵机接口方向



认识舵机设置指令

设置舵机引脚 S0 输出角度 90

输出角度取值范围为 $0^\circ \sim 180^\circ$

示例 16-1：舵机摆动

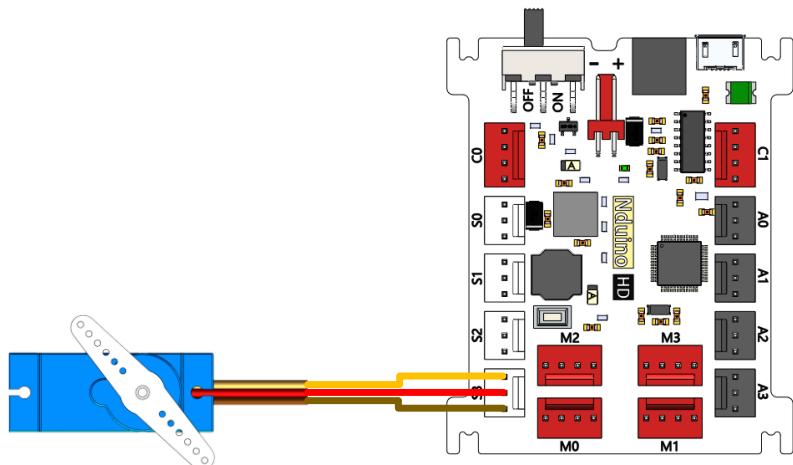
舵机从 0° 到 180° 之间反复摆动

元器件列表：

1. Nduino HD 主控板 $\times 1$

2. SG90 舵机 $\times 1$

电路连接：



程序编写：

好好搭搭硬件程序
重复执行
设置舵机引脚 S2 输出角度 0
等待 1 秒
设置舵机引脚 S2 输出角度 180
等待 1 秒

第十七节 红外遥控

红外遥控是广泛应用与生活电器的远程控制方式，需要一个红外信号发射器，一个红外信号接收器。

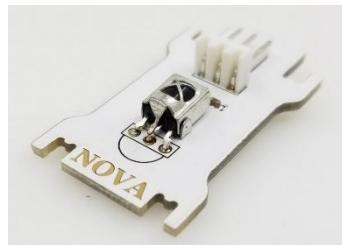
认识红外遥控器

红外信号发射器，简称“遥控器”，不同的按键按下，会发送不同信息，称为键值。



认识红外接收模块

利用红外接收模块，可以制作一个红外信号接收器，用于接收遥控器发出的信号。



认识读取红外值指令

读取红外值在引脚 S0

读取红外值指令相对于其他输入指令，不同之处在于，它不能直接放入诸如逻辑比较指令、数码管显示指令或串口输出指令中，必须先用变量将其保存起来。



认识红外按键值指令

红外按键值 IR_BUTTON_OK

红外按键值指令是为了方便用户使用制作的图形化指令，可以通过下来菜单来选择对应的按键，使用方法详见示例 19-1。

注意：该指令只支持官方配套遥控器，其它遥控器的使用方法详见 19-2。

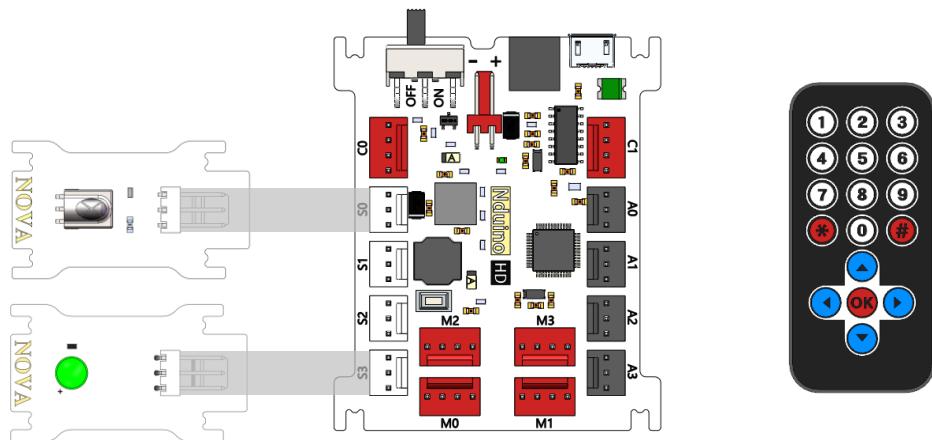
示例 17-1：红外遥控 LED

用红外遥控方式控制 LED 的亮灭

元器件列表：

1. Nduino HD 主控板 × 1
2. 红外接收模块 × 1
3. LED 模块 × 1
4. 红外遥控器 × 1
5. 3Pin 2510 连接线（白）× 2

电路连接：



程序编写：



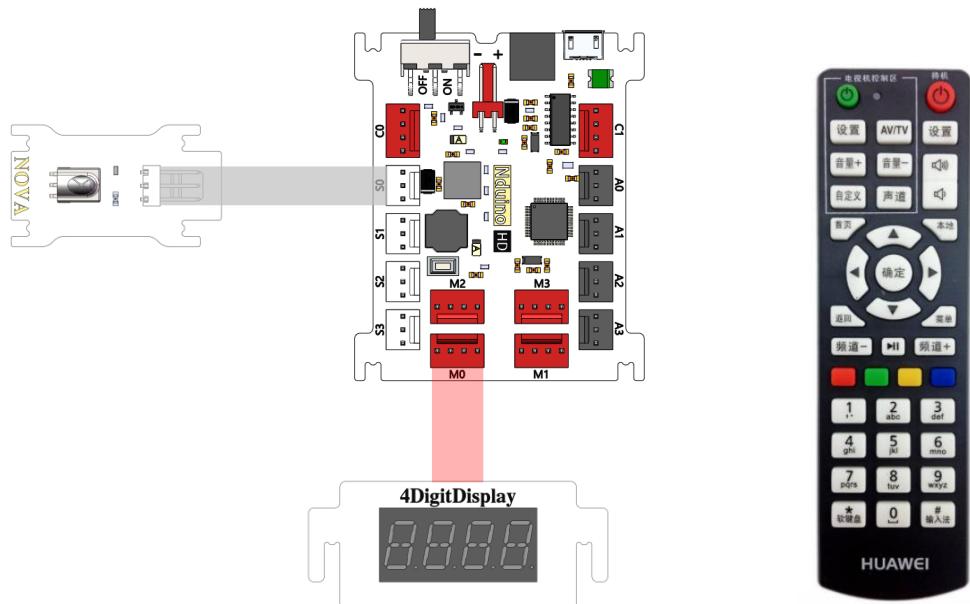
示例 17-2：用数码管显示红外遥控器的键值

红外接收模块除了支持官方配套遥控器外，也支持大多数生活中的遥控器，如电视遥控器、空调遥控器、投影机遥控器。

元器件列表：

6. Nduino HD 主控板 ×1
7. 红外接收模块 ×1
8. 数码管模块（计数）×1
9. 红外遥控器 ×1
10. 3Pin 2510 连接线（白）×1
11. 4Pin 2510 连接线（红）×2

电路连接：



程序编写：



当没有操作红外遥控器时，红外接收模块返回的一直是 0；而当按下红外遥控器上的按键，红外接收模块会接收到有且只有一个值，之后就算一直按住按键不放，红外接收模块也不会继续接收到新的值，只会返回 0，所以按键值就一闪而过，数码管上只能看到数字 0。

所以用 a 大于 0 的条件将按键值留下显示，其它的全部过滤掉。

通过上述方法，把遥控器的各个按键的值记录下来，如本例中用的某款电视遥控器中的“确定”键的值为 157，“返回”键的值为 295，利用这两个按键控制 LED 的亮灭，示例 19-1 的程序变为：

好好搭搭硬件程序

重复执行

将 **a** 设定为 读取红外值在引脚 S0

如果 **a = 157** 那么

设置数字口 S3 输出为 0

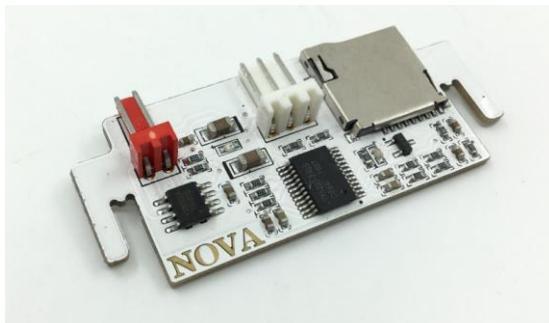
如果 **a = 295** 那么

设置数字口 S3 输出为 1

第十八节 MP3 音乐播放

MP3 音乐播放器在生活中随处可见，被集成到各种各样的电子设备中，如：手机、平板电脑、汽车、智能手表。创意作品的制作中恰当的使用 MP3 播放模块能极大提升作品的表现力。

认识 MP3 模块



正面



背面

认识 MP3 配件



扬声器



microSD 卡



SD 卡读卡器

SD 卡存放 MP3 文件

SD 卡根目录下新建一个名叫“mp3”的文件夹，将 MP3 音乐文件存入。



mp3 文件的文件名必须为“XXXX+任意字符”，如“0001 坦克大战”、“1578 魂斗罗”。文件名的前四位数字即是该 MP3 文件的曲目编号，根据曲目编号可以播放指定的 MP3 文件。

我的电脑 > (G:) > mp3

| 名称 | 修改日期 | 类型 | 大小 |
|--------------|-----------------|--------|-------|
| 0001坦克大战.mp3 | 2017-3-14 16:13 | MP3 文件 | 39 KB |
| 0002功夫.mp3 | 2017-3-14 16:06 | MP3 文件 | 56 KB |
| 0003拳皇.mp3 | 2017-3-14 16:08 | MP3 文件 | 38 KB |
| 0004超级玛丽.mp3 | 2017-3-14 16:10 | MP3 文件 | 55 KB |

认识 MP3 操作指令



MP3 操作指令共有 play、pause、stop、loop_play、random_play、next_song、last_song、vol_up、vol_down 九种操作模式。

(1) “play（播放）”模式：继续播放 MP3 文件。如 MP3 模块处于暂停状态，运行“播放”程序，则继续播放；如 MP3 模块处于停止状态，运行“播放”程序，则重头开始播放。

(2) “循环播放”模式：重头开始按顺序播放 MP3 文件，当最后一首 MP3 文件播放结束后，又从第一首开始播放。不论 MP3 模块处于什么状态，运行“循环播放”程序，则重头开始播放。

(3) “随机播放”模式：重头开始乱序播放 MP3 文件。不论 MP3 模块处于什么状态，运行“随机播放”程序，则播放第一首 MP3 文件，之后按乱序播放 MP3 文件。

(4) “暂停”程序：让播放暂停，下一次运行“播放”程序，则从暂停位置继续播放。

(5) “下一曲”和“上一曲”：切换歌曲播放。当“顺序播放”时，按顺序切换歌曲；当“随机播放”时，按乱序切换歌曲。

(6) “音量加”和“音量减”：调节音量。

(7) “停止”程序：停止播放，下次播放重头开始。

认识 MP3 曲目播放指令



该指令可以直接选取对应曲目文件播放，如参数为“1”，则播放文件名开头为“0001”的文件，如参数为“13”，则播放文件名开头为“0013”的文件。

示例 18-1：红外遥控 MP3 播放器

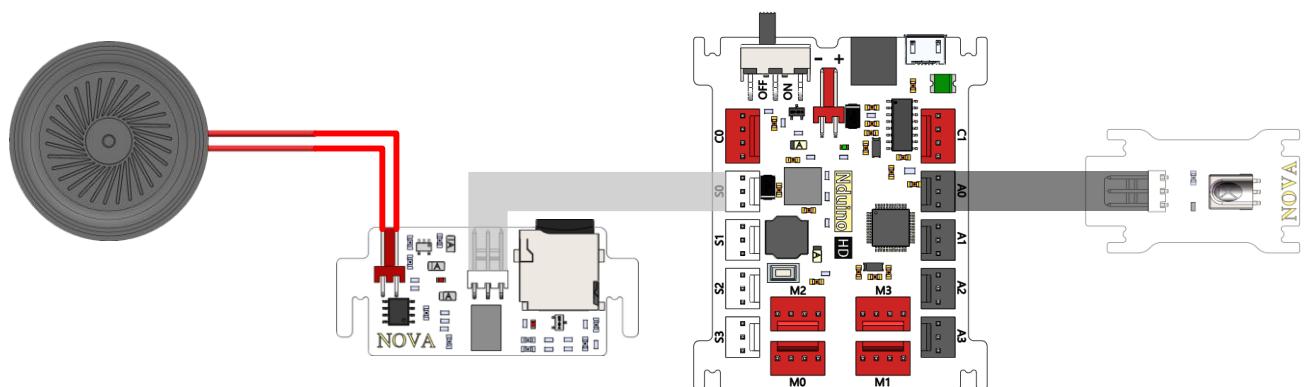
红外遥控器按键对应的 MP3 播放功能：

| 按键 | 模式 | 按键 | 模式 |
|-----|--------|----|------|
| 1~9 | 播放对应曲目 | OK | 播放 |
| * | 随机播放 | # | 循环播放 |
| ↑ | 音量加 | ↓ | 音量减 |
| → | 下一曲 | ← | 上一曲 |
| 0 | 停止 | | |

元器件列表：

1. Nduino HD 主控板 × 1
2. MP3 模块 × 1
3. microSD 卡 × 1
4. 扬声器 × 1
5. 红外接收模块 × 1
6. 红外遥控器 × 1
7. 3Pin 2510 连接线（白）× 2

电路连接：



程序编写：



播放控制相关程序



曲目选择相关程序

完整程序参见链接：<http://www.haohaodada.com/show.php?id=512899>。

第十九节 炫酷点阵屏

点阵屏在生活中随处可见，商业中心的外墙大屏幕、商店招牌、火车站候车信息等等。创意作品中恰当的使用点阵屏同样能极大的提升作品的表现力。

认识点阵模块

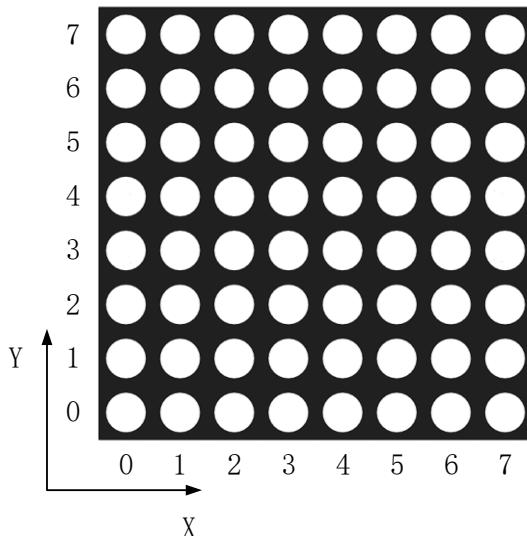
点阵模块实际上集成了很多个 LED 的模块，每个 LED 都有一个唯一的位置坐标。



正面



背面



认识点阵选点指令

Matrix drawPixel x 0 y 0 at C0▼

通过该指令中输入的坐标值可以选取要点亮的点。

认识点阵显示指令

Matrix writeDisplay at C0▼

单独运行选点指令不能直接点亮点阵模块，必须运行点阵显示指令之后才能点亮。

认识点阵清除指令

Matrix clear at C0

运行该指令清除点阵模块中的所有数据，熄灭 LED，同样需要配合点阵显示指令使用。

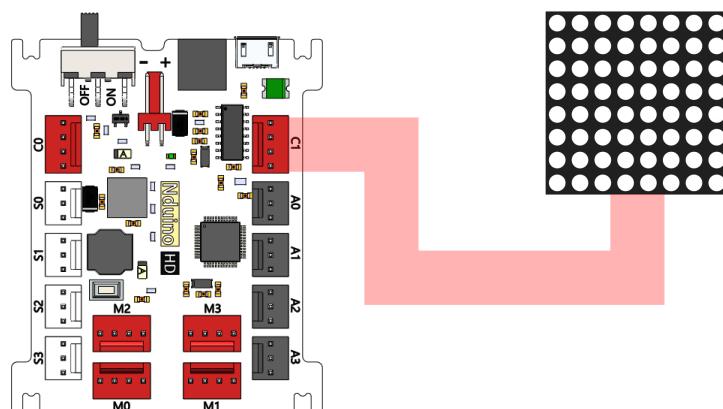
示例 19-1：按坐标点亮点阵模块

一行一行的把点阵点亮

元器件列表：

1. Nduino HD 主控板 × 1
2. 点阵模块 × 1
3. 4Pin 2510 连接线（红）× 1

电路连接：

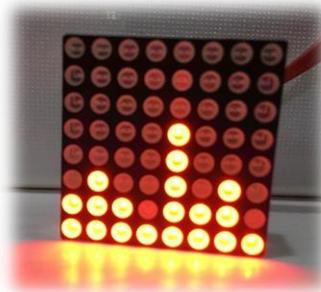


程序编写：



示例 19-2：音乐动感点阵屏——柱状图

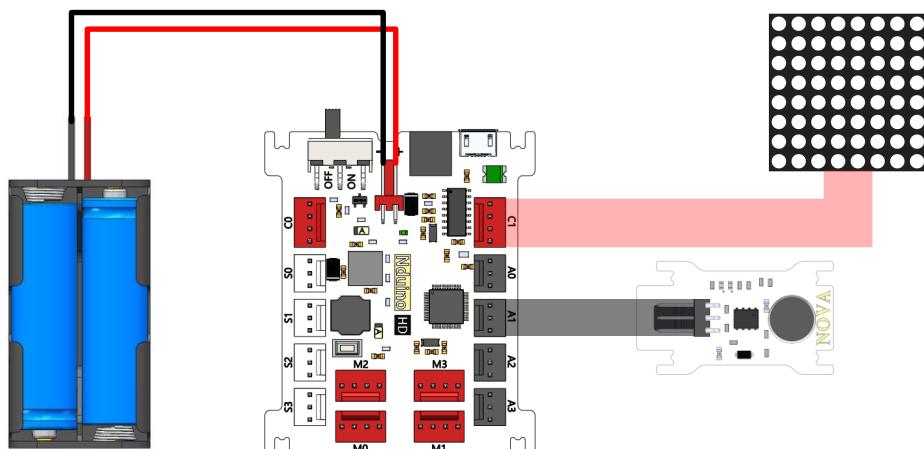
利用声音传感器采集声音信号，将音乐的律动显示到点阵屏上。



元器件列表：

1. Nduino HD 主控板 × 1
2. 点阵模块 × 1
3. 声音传感器模块 × 1
4. 3Pin 2510 连接线（黑）× 1
5. 4Pin 2510 连接线（红）× 1

电路连接：



程序编写：

打开音频播放器，当然也可以把 MP3 模块集成进去用于发声。先用串口助手观察声音的波形情况。



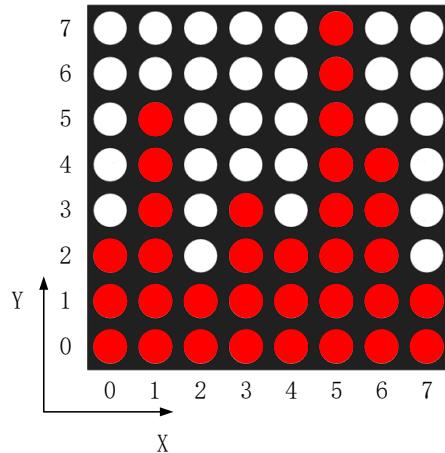


安静时



播放音乐时

从波形看，最大值约为 2000，最小值约为 400。正好分为 8 个区间，与点阵模块 Y 坐标的 8 个点对应。如声音值=535，则对应点亮 Y=0 和 Y=1 的点；若声音值=1800，则对应点亮 Y=0~8 的点。



从上图可以看出：

- (1) 点阵点亮的顺序是从左到右按列的点亮；
- (2) 每列都从 Y 坐标为 0 开始，一直点亮到峰值为止。

所以音乐动感点阵屏的程序如下：



第二十节 RTC 时钟模块

经历了之前课程的几十个程序编写之后，同学们有没有发现一个现象：电路系统在断电重启之后，所有的数据都会被重置，断电之前的程序运行结果都会丢失。

如果一个闹钟，每次断电之后都要重新调节时间，是不是太麻烦了？RTC 时钟模块可以断电存储时间数据，并自动计时。

认识 RTC 时钟模块



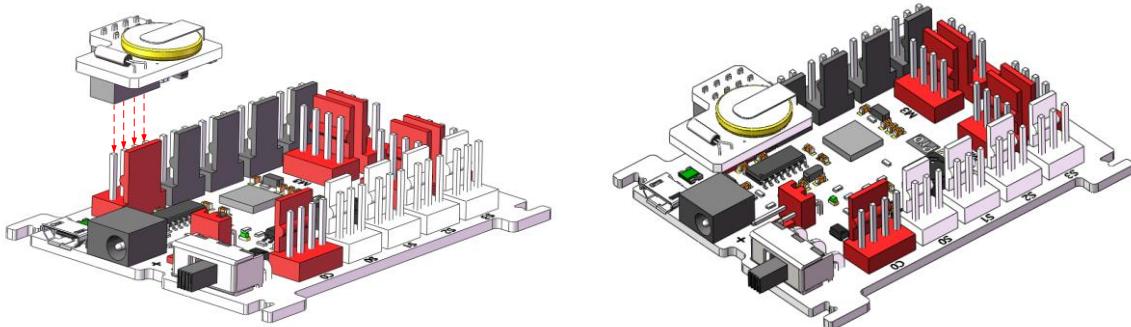
正面



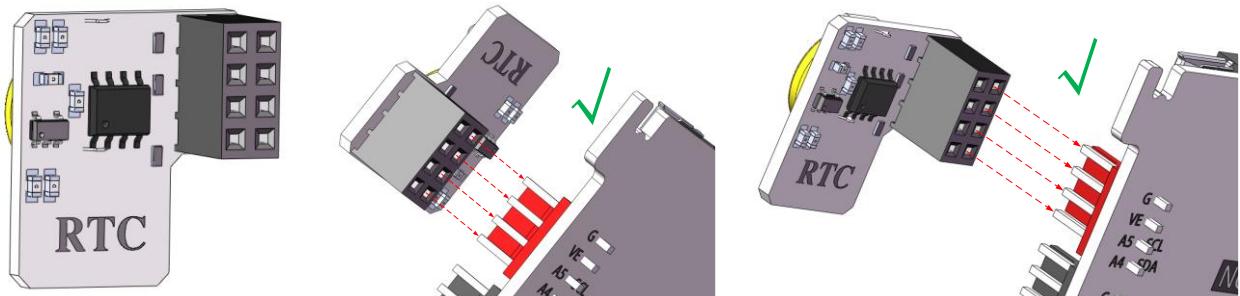
背面

RTC 时钟模块的安装

因为 RTC 时钟模块在结构上放在任何位置，对功能都没有影响，所以直接接插在主控板上是最简单的方式。



为了防止反接导致的危险，RTC 时钟模块设计了双排 4 针插口，可以接入任意一排 4 针插口。



认识 RTC 时钟读取指令



从 RTC 时钟模块中读取时间数据，可选 Second (秒)、Minute (分)、Hour (时)、Week (星期)、Day (日)、Month (月)、Year (年)。

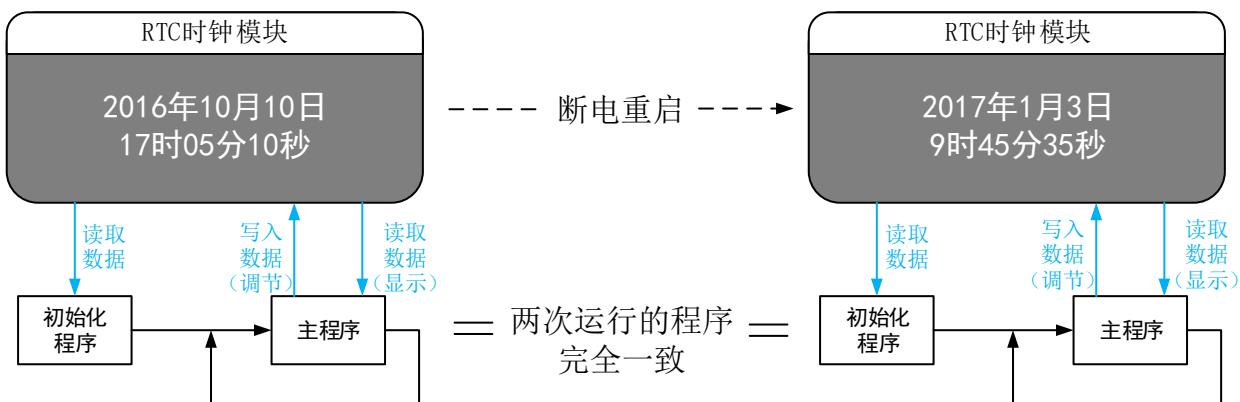
认识 RTC 时钟设置指令



将时间数据存入 RTC 时钟模块。

使用 RTC 时钟模块的基本逻辑

(1) 每次断电上电之后，程序都会全部重新运行，包括初始化程序。初始化程序中不能出现任何直接将数据写入 RTC 时钟模块的程序。否则每次断电重启之后，RTC 时钟模块中的自动运行的数据都会被程序给定的值覆盖。



(2) 每次调节时间之后，都必须将新的时间数据写入 RTC 时钟模块，否则更新的数据在断电重启之后会丢失。

示例 20-1：简易时钟

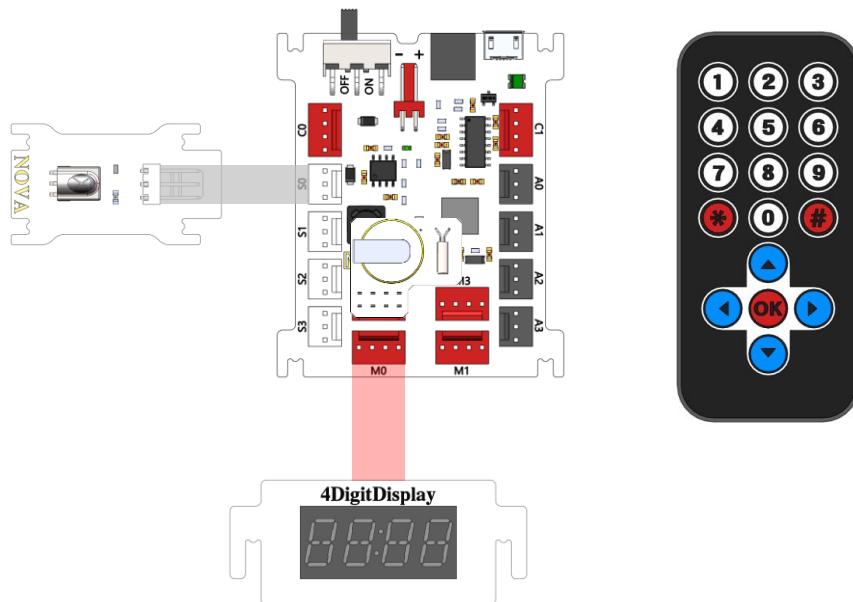
本例时钟是通过一个程序写入时间数据实现时间的设置，在通过另一个程序读取时间数据实现时间的显示。

元器件列表：

1. Nduino HD 主控板 × 1
2. 数码管模块（计时） × 1
3. RTC 时钟模块 × 1
4. 红外接收模块 × 1

5. 红外遥控器 × 1
6. 3Pin 2510 连接线（白）× 1
7. 4Pin 2510 连接线（红）× 1

电路连接:



程序编写:

时间设置程序:



时间显示程序:

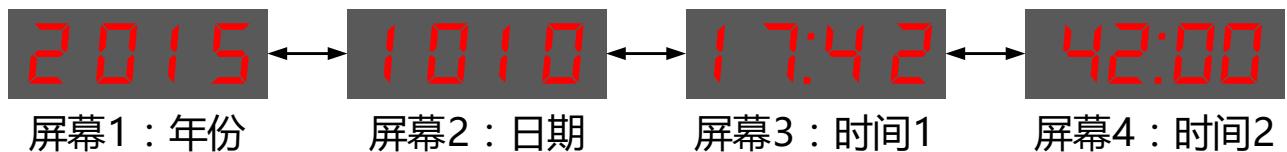
数码管只能显示 4 位数字，所以年份、日期、星期、时刻 1 (时和分)、时刻 2 (分和秒)，必须分屏出现，所以做如下定义：

屏幕 1：显示年份

屏幕 2：显示日期

屏幕 3：显示时刻 1（时和分）

屏幕 4：显示时刻 2（分和秒）



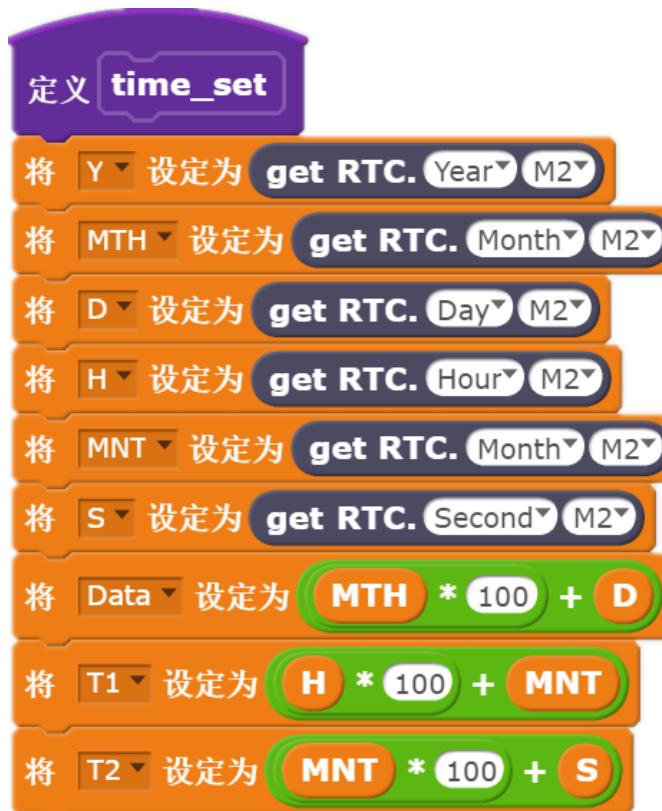
利用遥控器上的方向键左和右进行屏幕切换。

(1) 初始化程序为：



其中，变量 mode 为屏幕号，初始化为 4，即默认显示时间 1（时和分）。

(1) 主程序模块 1——时间数据更新程序



其中，

- 各时间变量声明的赋值，全部从 RTC 时钟模块中读取对应数值。
- 变量 Date=month×100+day，是为了让月份和日期同屏显示，如 11 月 9 日，数码管应显示为 1109=11×100+9。变量 T1 和 T2 同理

(2) 主程序模块 2——屏幕模式选择程序



(3) 主程序模块 3——遥控器操作程序



最后，把所有程序放入主程序中。



到此，简易时钟程序编写完毕。本案例需时间设置和时间显示两个程序，不能直接调整时间。同学们可以思考下如何利用遥控器实现调节时间的功能。

第二十一节 蓝牙远程控制

如今，人们已经习惯了手机应用带来的各种便利。做了那么作品，大家对通过手机 APP 控制自己制作的智能硬件作品都有着不小的冲动吧！这节课，将带大家实现一个简单的手机蓝牙 APP 远程控制 LED 亮灭的应用。

认识蓝牙模块

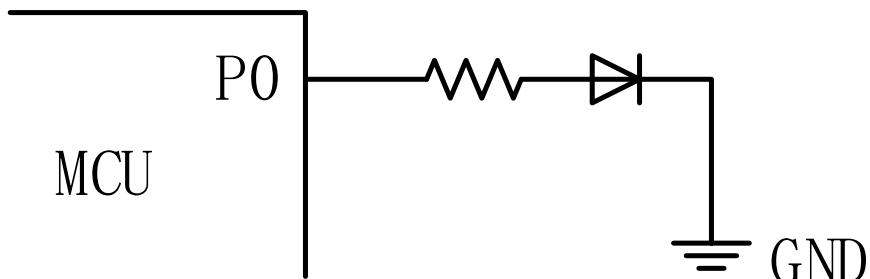


蓝牙模块的安装

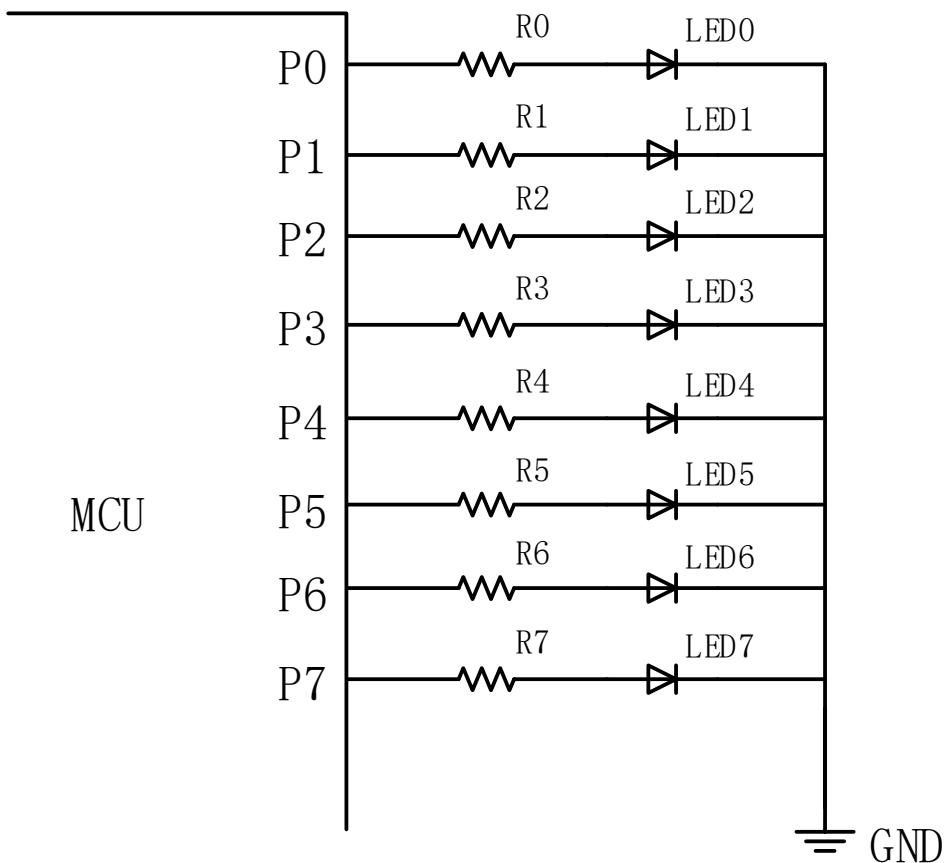
蓝牙模块的安装与 RTC 一样，是可以直接插到主控板上的。不同的是 RTC 时钟模块可以插在主控板任何一个红色接口上，而蓝牙模块只能接在 c0 上。

认识蓝牙通讯的原理

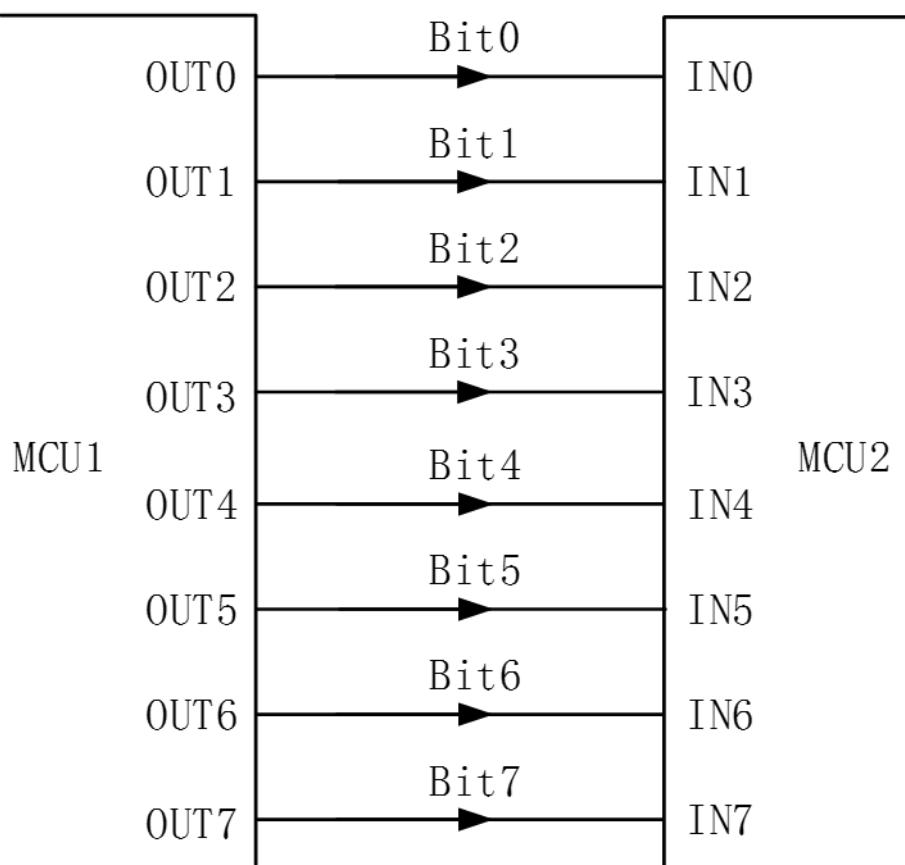
在“第十四节 串口打印”中介绍了串口的相关应用。这里对串口通讯做更详细的介绍。



上图是 MCU（微控制单元）驱动单个 LED 的电路，那么驱动多个 LED 的电路呢？玩过 Arduino 朋友都不会陌生。如下图：

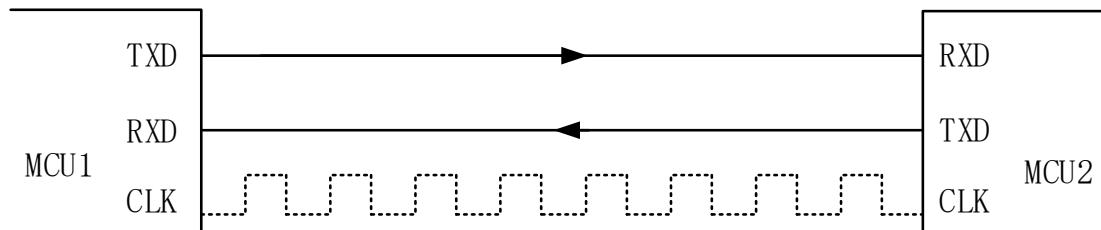


像上图这样由多个 IO 口同时传输数据的通讯方式，称为“并口通讯”。



形如上图的通讯方式均为“并口通讯”。不难想到并口通讯方式的优点是传输速度快，而缺点是线路复杂，一次要传输多少位的数据，就得连接多少根导线。

而“串口通讯”，通常只有“TXD（发送）”和“RXD（接收）”两根信号线，不论有多少数据需要传输，都通过这两根线实现，接线简单。

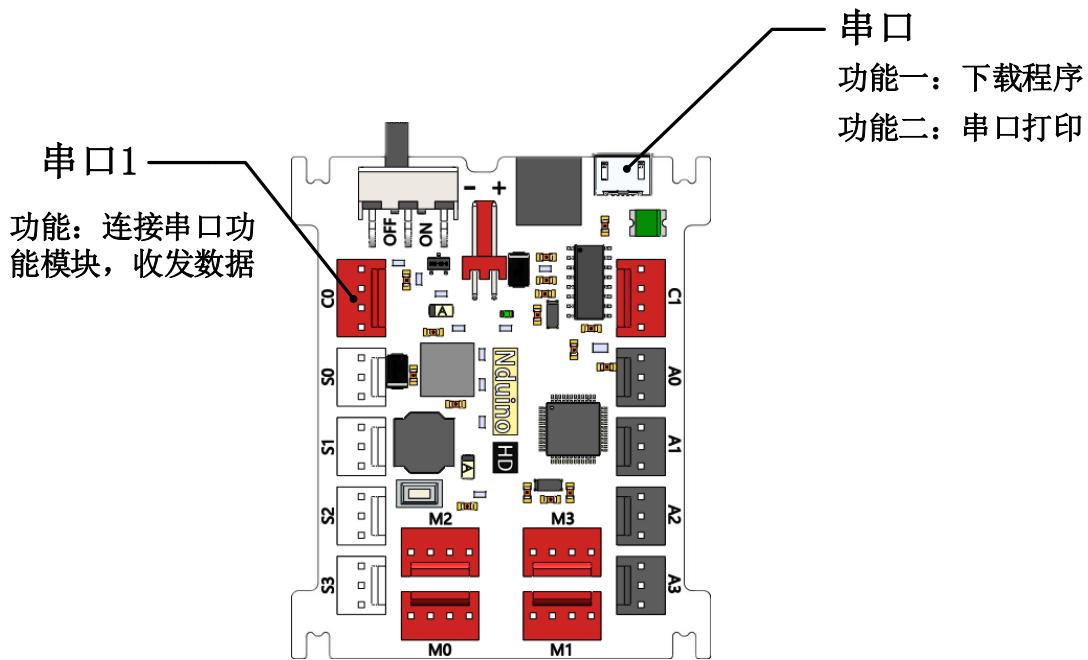


其中 CLK 不是一根导线，它是“波特率”，指通过串口发送数据的速度，即单位时间内发送数据的个数。互相传输数据的两端的波特率设置必须完全相同，才能保证通讯的正常。

“一汽车倒车，一路人很热心——“倒……倒……倒……倒不得了！”可车子一只轮胎已滑进路边水沟。车夫怒气冲冲下来，旁观者说，那人是一结巴。”

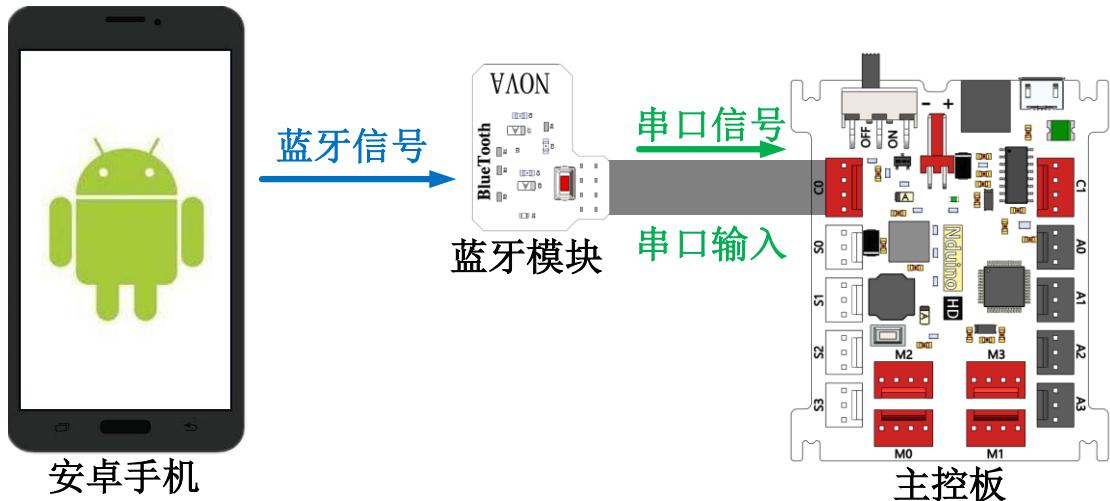
通讯两端波特率不同就会闹出像上面笑话里那样的误会。

NOVA HD 主控板上的串口有两个，分别为“串口”和“串口 1”，对应 USB 接口和 C0 接口。USB 串口负责下载程序和串口打印；C0 串口 1 则用来连接蓝牙模块。

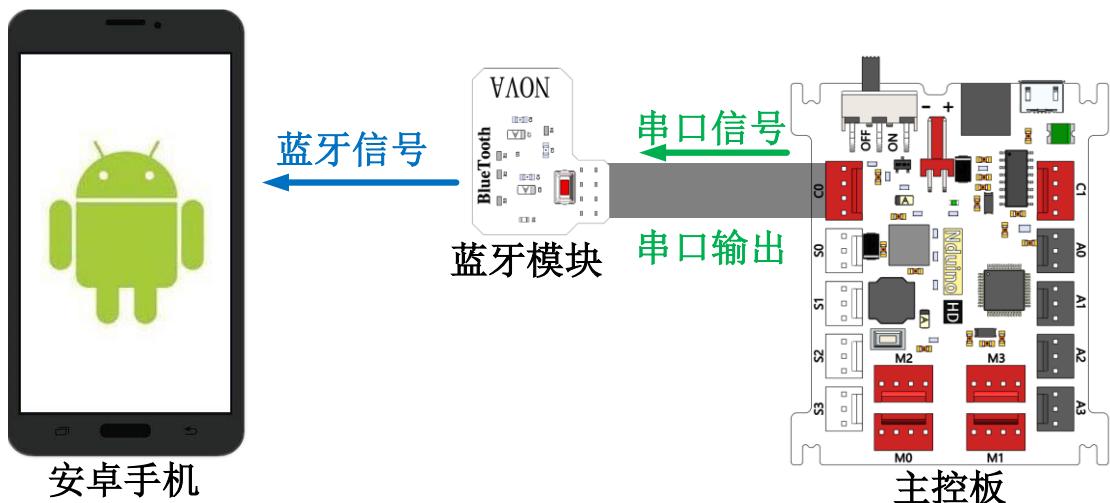


这里有的同学可能会问，为什么不叫“串口 1”和“串口 2”？是因为其命名规则沿用 Arduino 体系下的“Serial”和“Serial1”。

手机 APP、蓝牙模块、NOVA HD 主控板三者的连接和数据流向是怎样的呢？



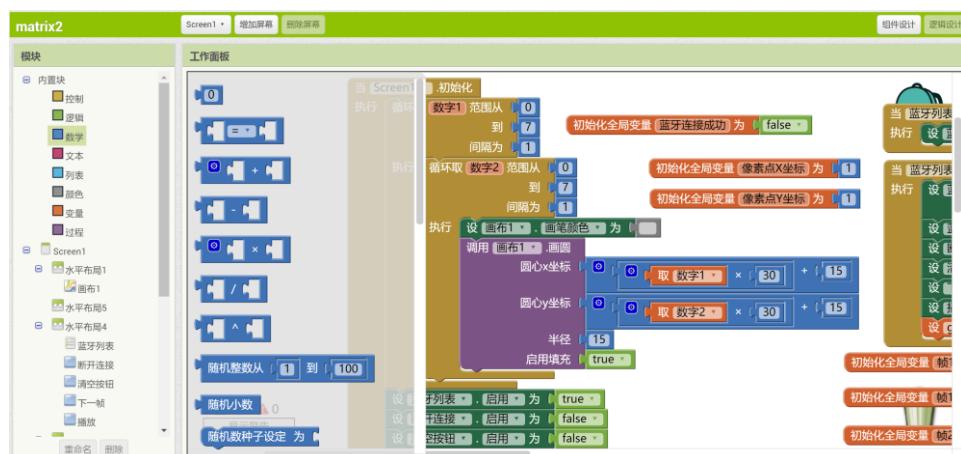
手机 APP 向 NOVA HD 主控板发送数据



NOVA HD 主控板向手机 APP 发送数据

认识 App Inventor

App Inventor 是一款谷歌公司开发的手机 APP 编程软件，与是类似于 Scratch、Mixly 的图形化积木式编程软件。



APP Inventor 也有在线编程平台，这里笔者推荐广州电教网的：<http://app.gzjkw.net>

APP Inventor 的相关资料网站推荐：<http://www.17coding.net/>。

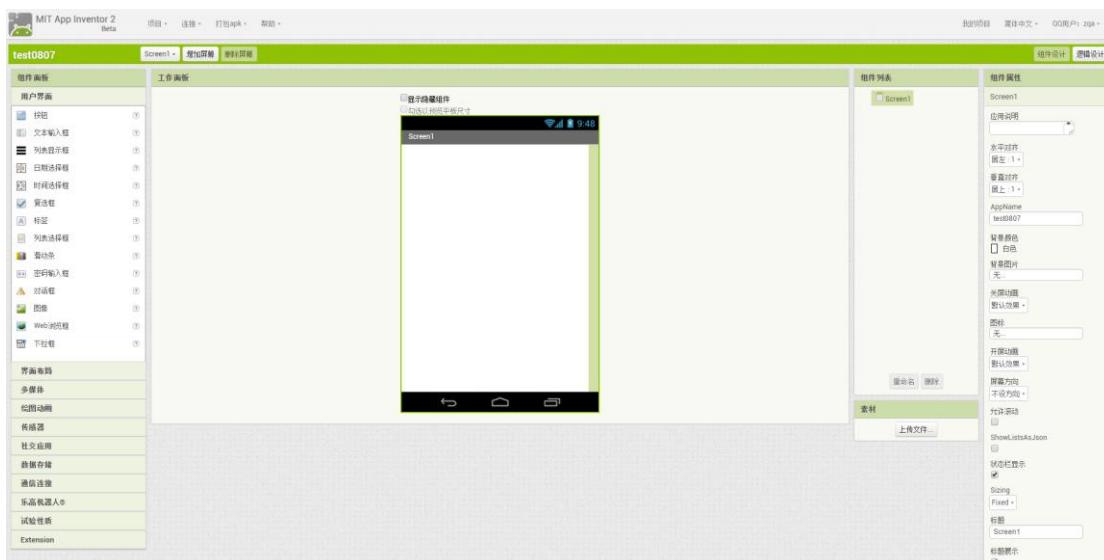
APP Inventor 因为没有官方中文版，所以各家中文版平台的翻译略有不同，查阅资料时发现命名上的冲突，最好是查找对应的英文说明。

登录 APP Inventor 编程平台



可以直接用 QQ 账号登录。

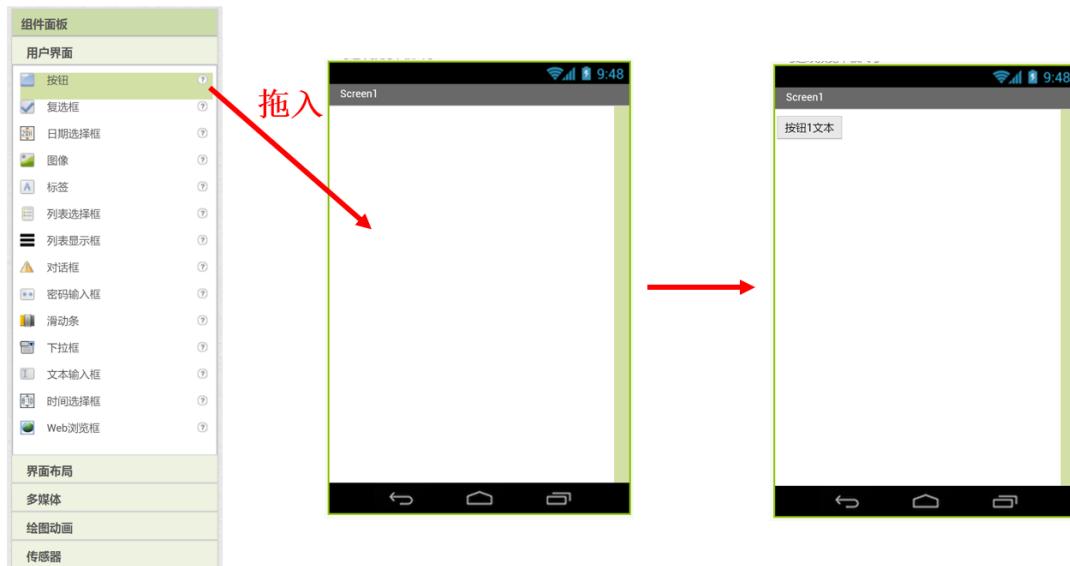
新建项目：



APP Inventor 的项目分为“组件设计”和“逻辑设计”两部分，上图是组件设计界面。

其中，“组件面板”是组件库，里面有按键、文本框、滑动条、画布、蓝牙客户端等一系列组件，可以通过鼠标拖拽的方式添加到 APP 中。

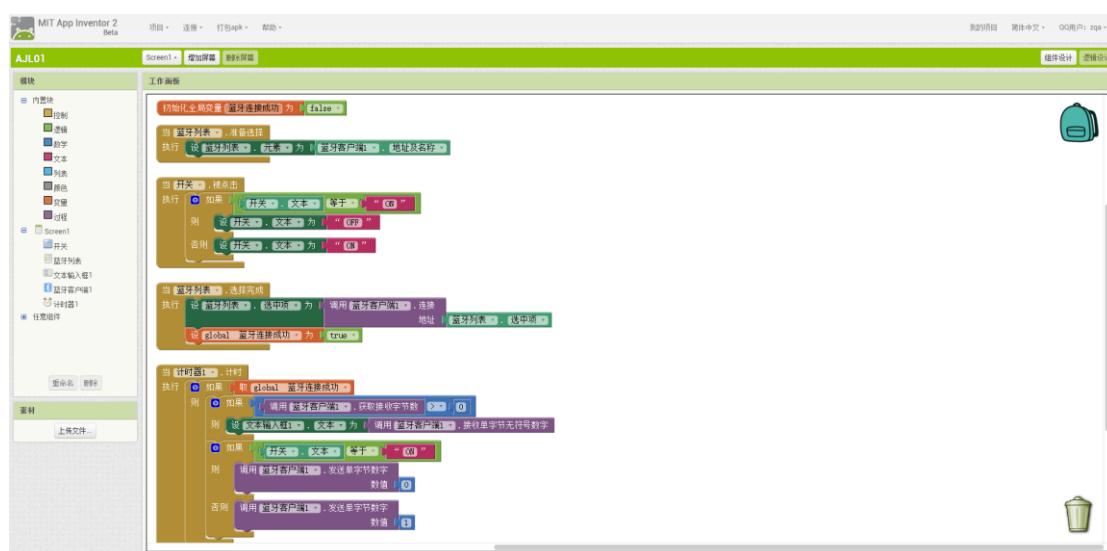
“工作面板”的内容和最后生成的 APP 是完全一致的，即可以通过工作面板看到你最后做出 APP 是什么样子的。



“组件列表”是 APP 中所有组件的关系树，即屏幕 1 中有几个组件、屏幕 2 中有几个组件，名称各是什么。

“组件属性”是各个组件的具体参数设置，大部分参数可以在逻辑程序运行过程中修改。

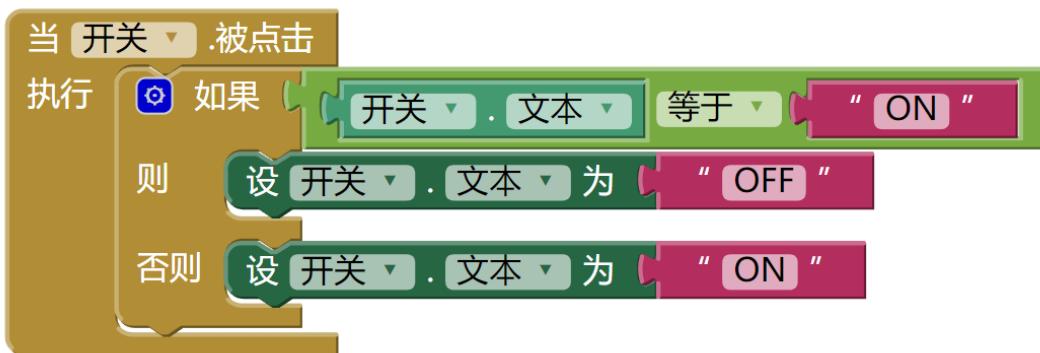
逻辑设计是用于设计 APP 各组件对应的程序，即设计当用户操作各组件时，APP 做出什么样的反应。



示例 21-1：蓝牙控制 LED

本次教程将通过 APP Inventor 编写一个控制 NOVA 端 LED 灯亮灭的程序。

在编写完整的 APP 之前，先实现按钮的一个小程序，即按一下按钮，按钮的文本由“ON”变为“OFF”；再按一下按钮，按钮的文本由“OFF”变为“ON”；



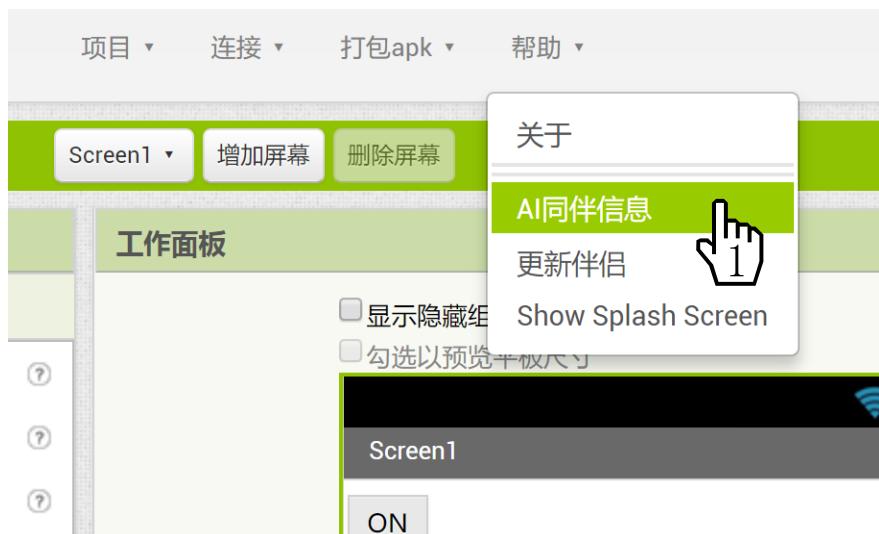
这些模块的颜色与 Scratch 相似，按照不同的功能种类，有不同的颜色，大家可以根据示例程序中各模块的颜色去模块库中寻找，这里不多做赘述。

在做完上述小程序 APP 后，可以让它在手机上运行。可以用 AI 伴侣快速的在手机上运行写好的 APP。

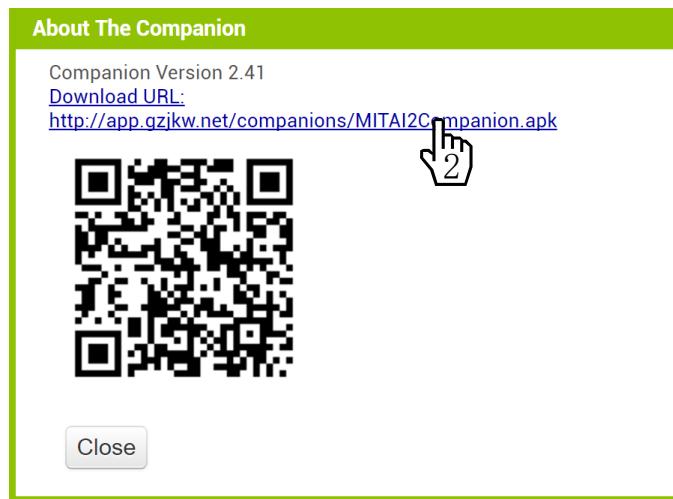


MITAI2Companion.apk

点击“帮助”菜单下的“AI 同伴信息”：



点击链接下载：



浏览器可能会弹出“不安全”提醒，选择继续访问。该网站是安全可靠的，由广州市电教馆开发维护。

手机上打开 AI 伴侣 APP，扫描网页上 AI 伴侣生成的二维码，即可快速的运行编写好的 APP，具体操作流程如下：



蓝牙程序的编写

蓝牙相关程序的编写，需要添加一个“蓝牙客户端”组件，它是非可视组件，即在 APP 中看不见它。

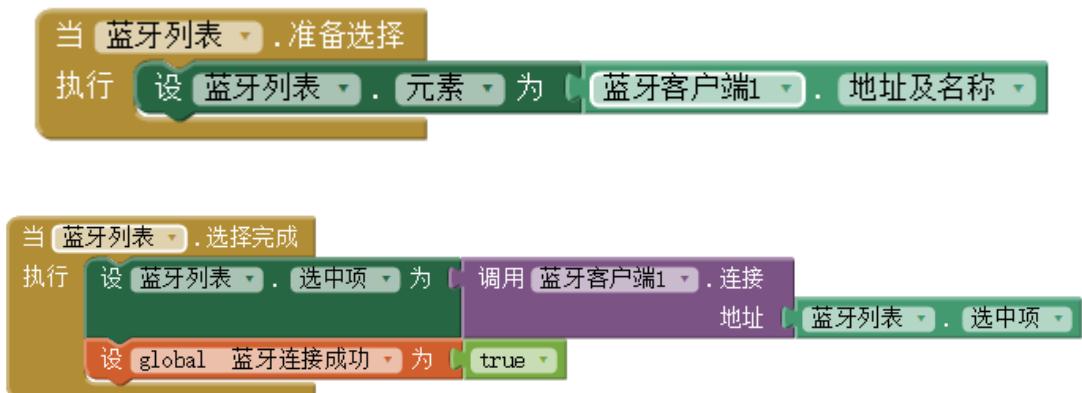


蓝牙模块需要选择之后才能连接，所以这里需要添加一个“列表选择框”组件。



将“列表选择框”组件名和文本名都改为“蓝牙列表”，提升程序可读性。

在逻辑设计界面里编写蓝牙模块选择程序。

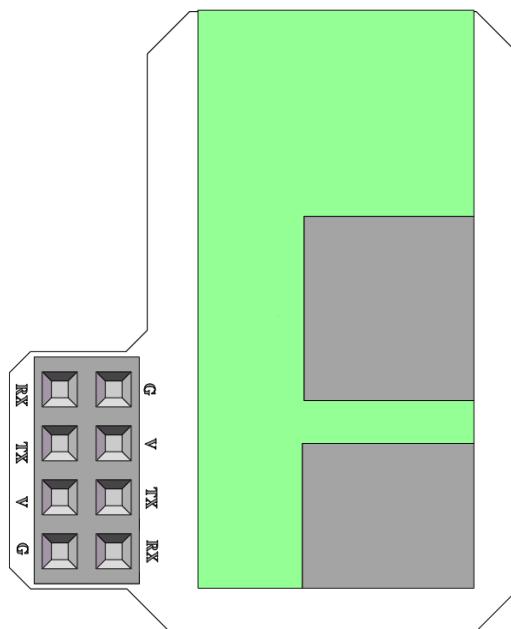


程序运行效果：点击“蓝牙列表”组件，弹出蓝牙地址的选择框；再点击选择要连接的蓝牙地址，之后蓝牙连接成功。

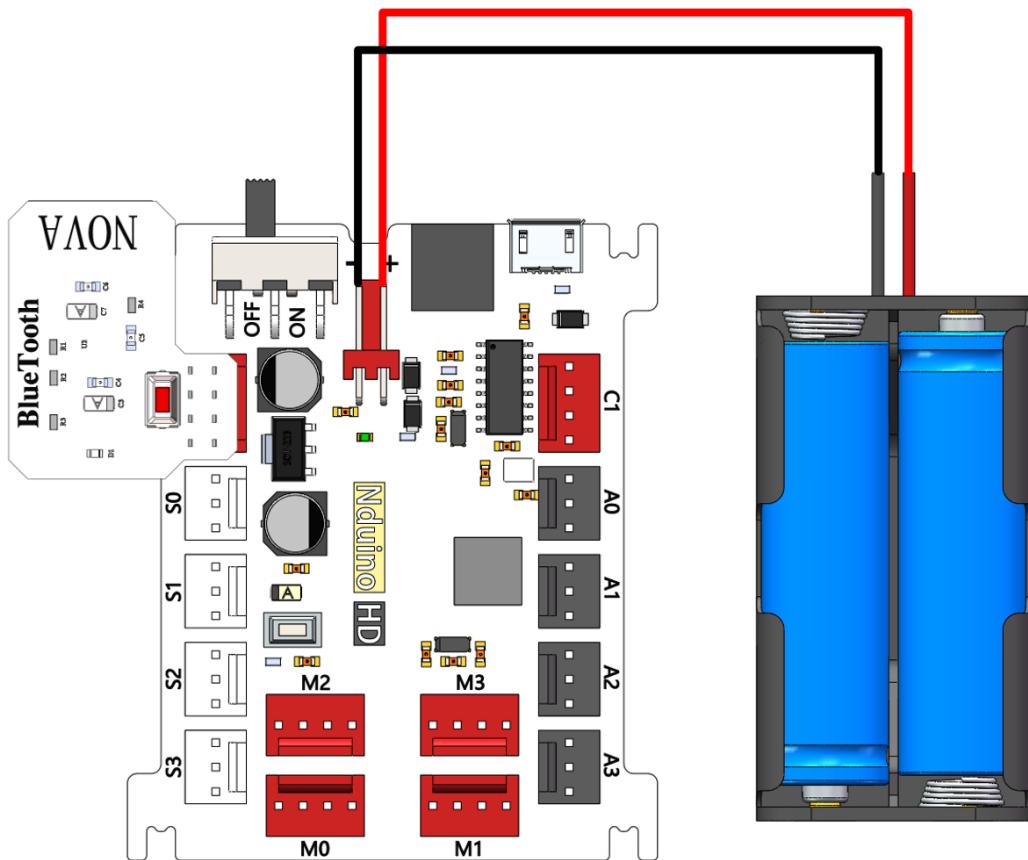
接下来配合 NOVA HD 主控板和蓝牙模块，来实现手机 APP 和蓝牙模块的连接。

NOVA HD 主控板与蓝牙模块的连接

NOVA 的蓝牙模块上有一个 8 针的接口



这里的设计是为了实现防反接，两列的 4 个接口是轴对称的，只要做到与 C0 4 对 4 的接插，就一定不会有问题是。



因为接的 C0 是红色接口，所以需外部供电，可以接上电池或适配器。

供电之后，蓝牙模块上的 LED 灯会闪烁，表示未连接，一旦连接成功，LED 将变为长亮。

手机与蓝牙模块的配对

这里用的蓝牙模块为蓝牙 2.0 模块，在连接之前，需要在系统设置中完成配对。

配对示例：（示例系统版本为 Android 5.0）



用 AI 伴侣，将写好的 APP 在手机上运行。按照下图的操作顺序，实现蓝牙模块的连接。



LED 长亮之后，代表蓝牙连接成功。接下去继续编写蓝牙程序，让按钮处于“ON”和“OFF”时发送不同的数值。



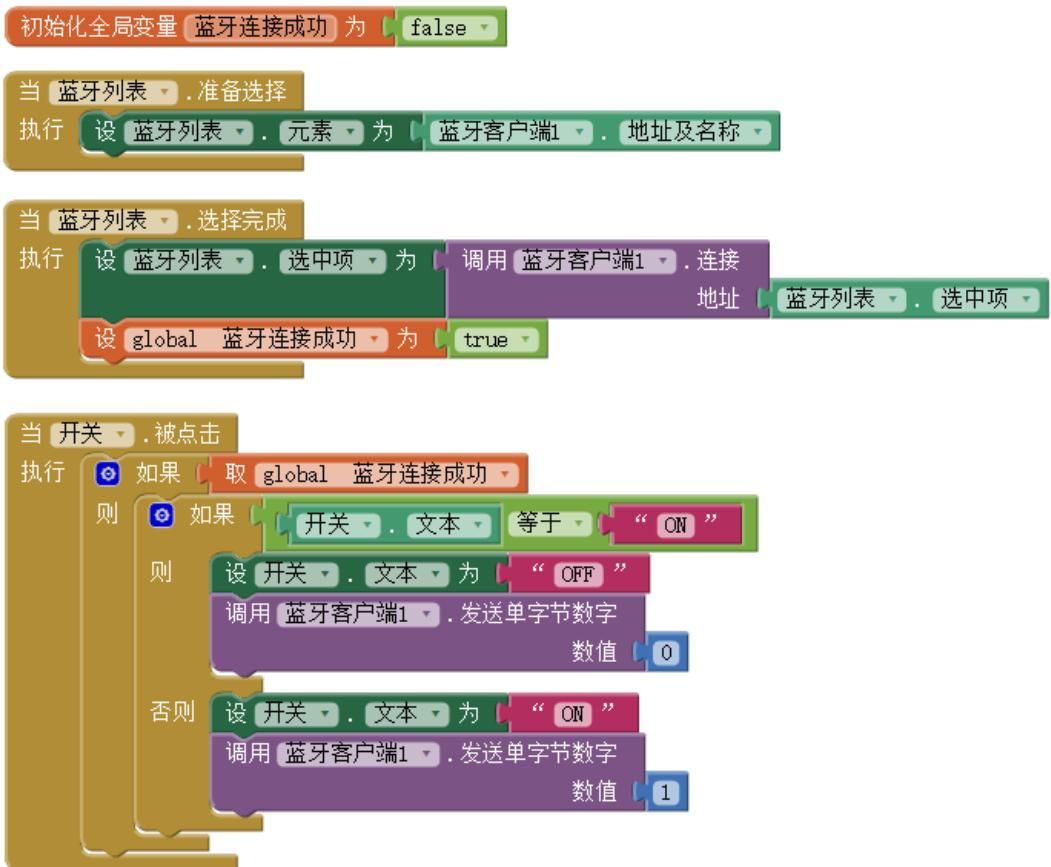
在按钮文本切换的程序中添加蓝牙客户端发送数字的程序模块。

这时运行 APP，如果未连接蓝牙，就电机“ON/OFF”按钮，会弹出一个错误提示，是因为蓝牙无连接情况下发送数字的报错。

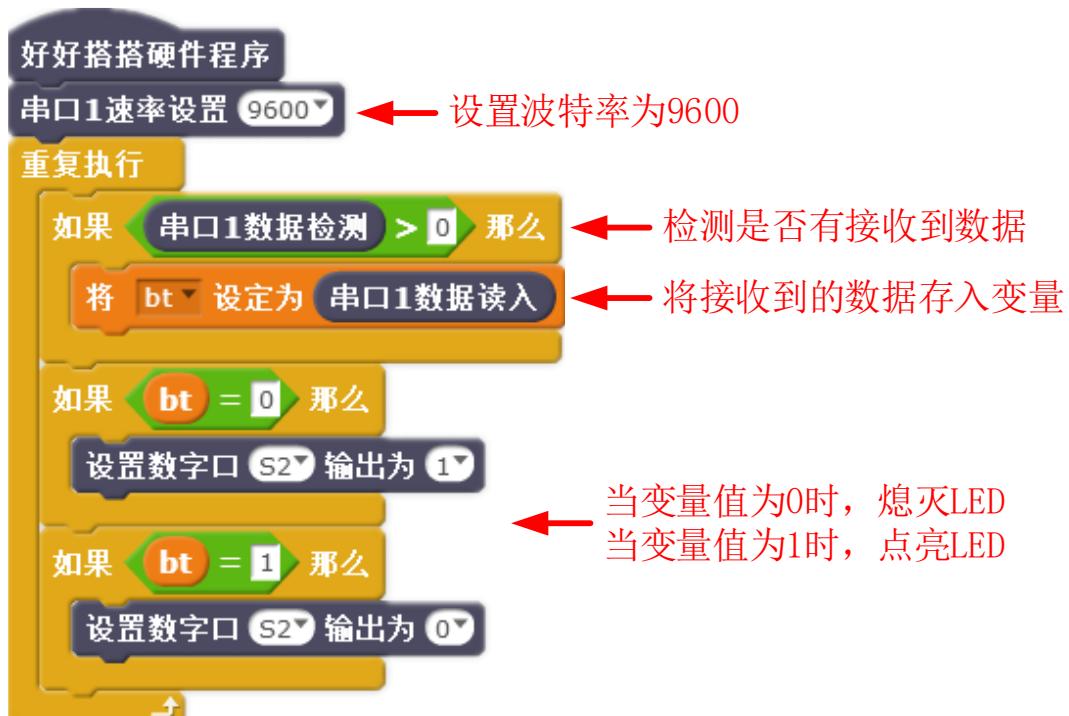


所以这里应该添加一个判断条件。即新建一个全局变量“蓝牙连接成功”，它是一个布尔量，只有“`true`（真）”和“`false`（假）”两个值可取。

初始化时，变量“蓝牙连接成功”为“`false`（假）”，当蓝牙地址选择完成之后，将其变为“`true`（真）”。



手机 APP 端的程序编写完成，接下来编写 NOVA HD 端的程序。



全部的编程工作完成，可以用手机控制 LED 的亮灭了，同学们试试看吧。

第四部分

软件进阶

第二十二节 计时器

之前的项目案例多为单一功能，不存在多个功能之间产生冲突的问题。如一个复杂应用中需要有 LED 闪烁这个小功能，如果使用第二节中介绍 LED 闪烁程序的方法实现，等待指令将导致整个程序的响应能力变的极差。这节的内容将介绍解决复杂程序中的冲突问题。

认识计时器指令

计时器

计时器的值由主控芯片自动产生，单位为毫秒，按时间自动增加，完全不占用程序运行的时间等待指令的运行会使整个程序停止，如果这时输入有改变是读取不到的。这是两者最大的区别。

计时器指令需要配合两个变量一起使用，取名为 t1 和 t2。其中 t1 负责读取系统运行时间，t2 用于与 t1 进行比较。

如希望某事件的时间间隔为 1 秒（1000 毫秒）：

- (1) 让 t1 的值与系统运行时间始终保持一致；
- (2) 当 $t1-t2$ 的值大于 1000（毫秒）时，则让 t2 的值更新为 t1 的值，否则不更新。

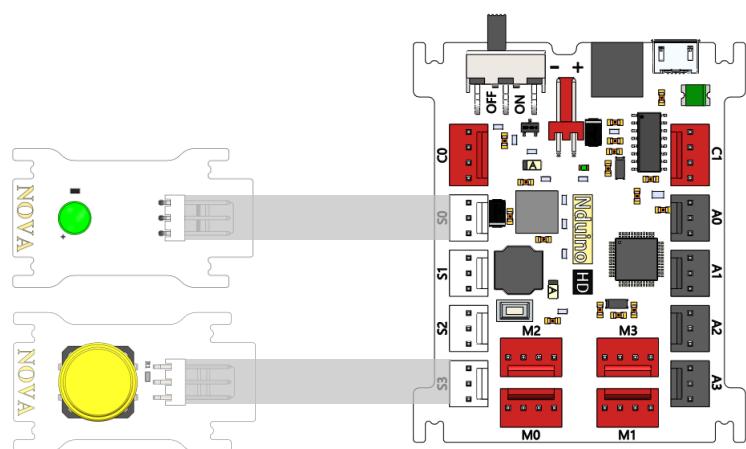
示例 22-1：长按点亮 LED，松开熄灭 LED

长按按键 3 秒以上才能点亮 LED，短按按键则马上熄灭 LED。

元器件列表：

1. Nduino HD 主控板 × 1
2. 按键模块 × 1
3. LED 模块 × 1
4. 3Pin 2510 连接线（白）× 2

电路连接：



程序编写：



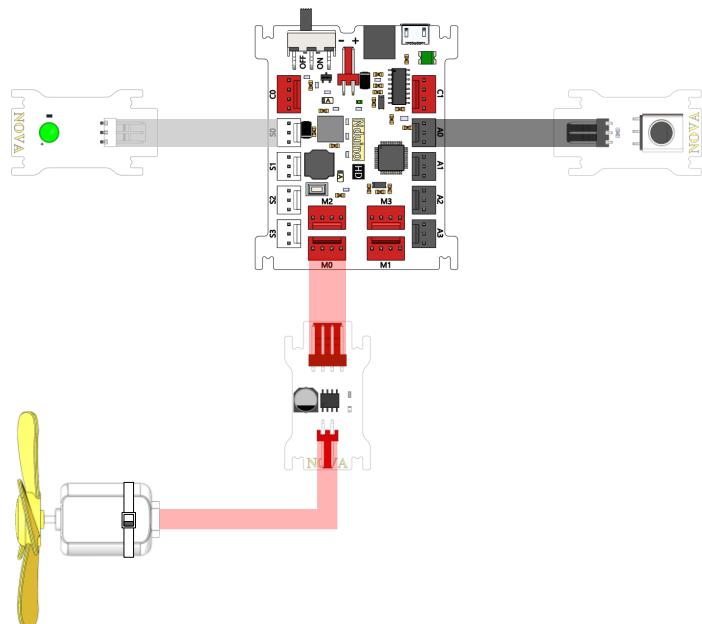
示例 22-2：LED 闪烁的同时，用电位器控制风扇的转速（等待指令）

用等待指令编写程序

元器件列表：

5. Nduino HD 主控板 × 1
6. 单路电机驱动模块 × 1
7. 130 电机 × 1
8. 风扇叶片 × 1
9. 电位器模块 × 1
10. LED 模块 × 1
11. 4Pin 2510 连接线（红）× 1
12. 3Pin 2510 连接线（白）× 1
13. 3Pin 2510 连接线（黑）× 1

电路连接:

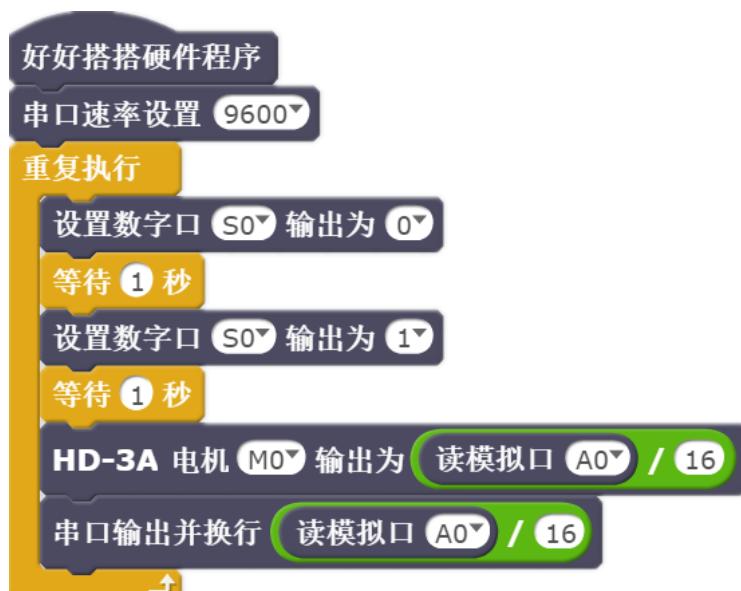


程序编写:

用等待指令实现 LED 的闪烁:



在加上电位器控制风扇的程序，同时用串口助手将电位器的值打印出来:



程序运行结果：

打开串口助手，观察电位器的值。



发现电位器的值每两秒打印一次，风扇转速变化也不平滑。出现这个结果正是因为等待指令运行时，电位器的变化是读取不到的。接下来用计时器指令实现 LED 闪烁试试看。

示例 22-3：LED 闪烁的同时，用电位器控制风扇的转速（系统运行时间）

用计时器指令编写程序

元器件列表：

同示例 24-2。

电路连接：

同示例 24-2。

程序编写：

用计时器指令实现 LED 的闪烁，需定义 mode、t1 和 t2 三个变量。

好好搭搭硬件程序

串口速率设置 9600

将 mode 设定为 0

—— 用于控制LED的亮灭。当为0时，LED灭；反之，LED亮

将 t1 设定为 计时器

—— 用于读取系统运行时间

将 t2 设定为 计时器

—— 用于与t1进行比较

重复执行

将 t1 设定为 计时器

—— 随着系统运行时间不断增加。

如果 t1 - t2 > 1000 那么

—— 当t1与t2的差值超过1000毫秒时。

如果 mode = 0 那么

—— mode的值“0/1”切换，并切换LED的亮灭状态。

将 mode 设定为 1

设置数字口 S0 输出为 0

否则

将 mode 设定为 0

设置数字口 S0 输出为 1

将 t2 设定为 t1

—— 将t2的值更新为t1，开始新一轮的计时。

HD-3A 电机 M0 输出为 读模拟口 A0 / 16

串口输出并换行 读模拟口 A0 / 16

程序运行结果：

打开串口监视器，观察电位器的值，这时可以看到打印值刷新非常快。电机转速的控制也非常流畅。



在复杂程序中，应避免使用等待指令，所有需要用到时间间隔的程序，都可以用计时器指令实现。

第二十三节 输入计数及其应用

要实现诸如记录按键按下次数，或通过按下松开按键一次切换输出器件状态，需要巧用变量来实现。

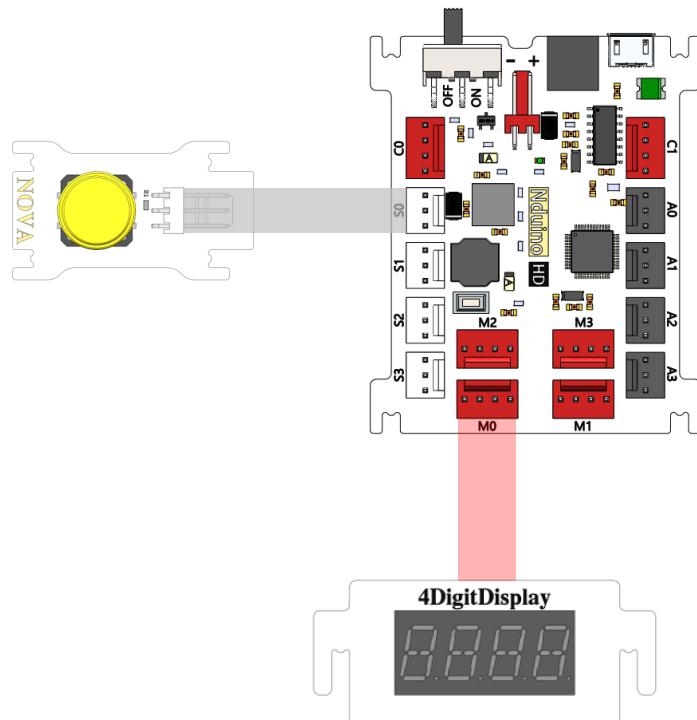
示例 23-1：记录按键按下松开的次数，并显示到数码管上

按下按键一次，不论按下多长时间，只要不松开，数码管上的数字只增加 1。

元器件列表：

1. Nduino HD 主控板 × 1
2. 按键模块 × 1
3. 数码管模块（计数） × 1
4. 3Pin 2510 连接线（白） × 1
5. 4Pin 2510 连接线（红） × 1

电路连接：



程序编写：

大家首先想到应该是下面这样的程序：

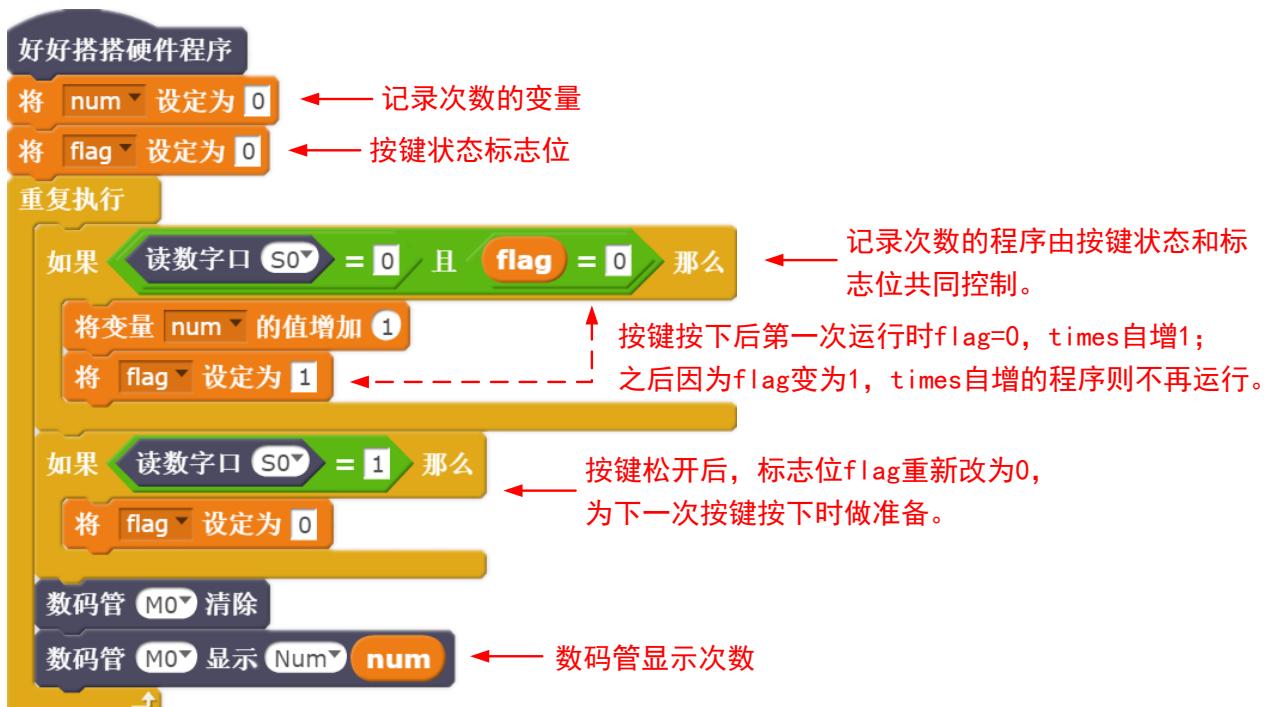


这个程序的运行结果是按下一次，数码管显示的数字增加的非常快。即使手速再快，每按一次，数码管显示的数值都会至少增加几十。

大家应该都能分析出程序错误的原因：按键被按下的时间里，程序运行了很多次，变量 num 自增了很多次。

所以，问题的关键在于按键被按下时，如何让计数程序只运行一次。

这需要声明一个变量，用于记录按键的松开和按下状态，工程上这样的变量被称为“标志位”。



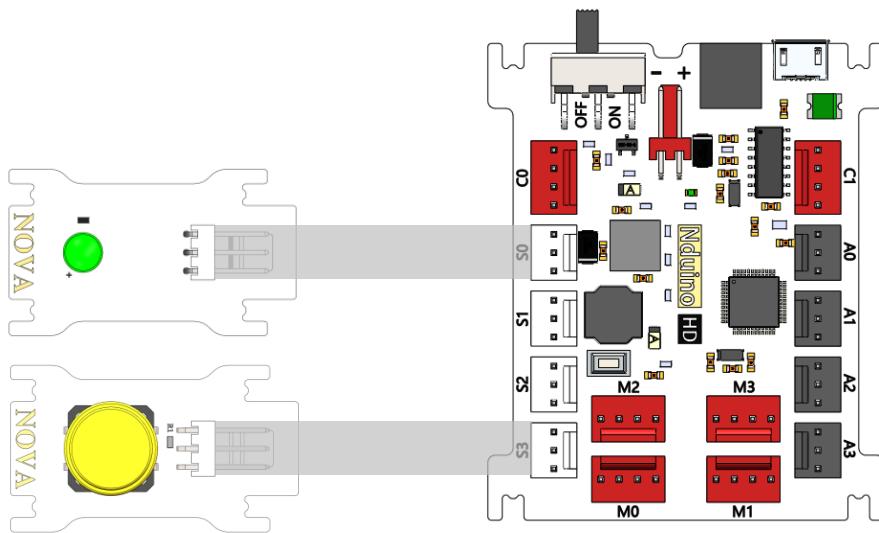
示例 23-2：点控 LED

按键按下松开一次，LED 亮；再按下松开一次，LED 灭；如此往复。使用重复程序块和使用延时程序块一样，会阻断程序的运行。接下来介绍的才是真正正确的办法。

元器件列表：

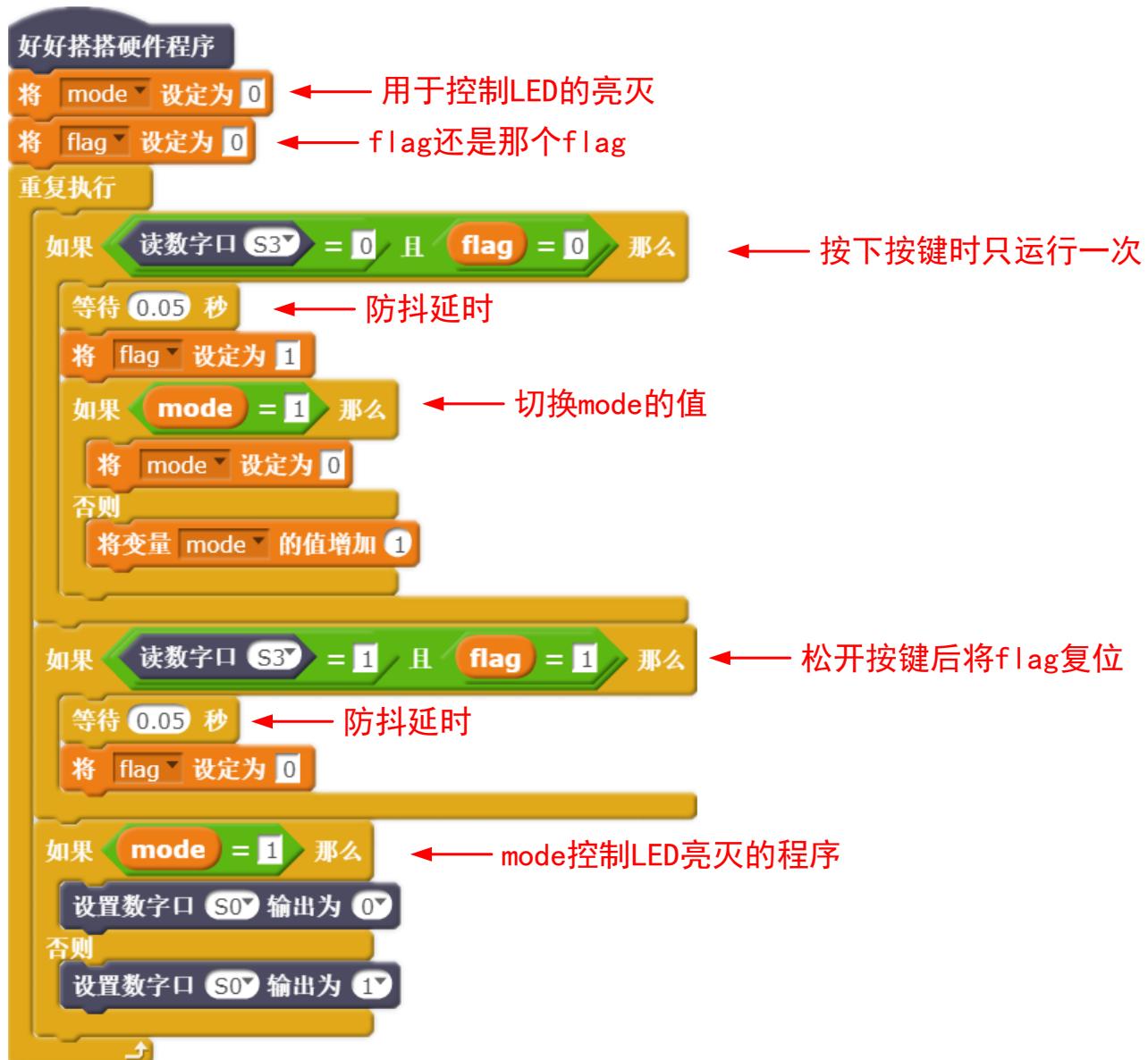
1. Nduino HD 主控板 × 1
2. 按键模块 × 1
3. LED 模块 × 1
4. 3Pin 2510 连接线（白）× 2

电路连接：



Mixly 程序编写:

根据示例 25-1，可以轻松的理解下面的程序：



其中，按键按下部分的程序如下：



这是更通用的方式，当按键控制的模式不只有亮和灭两种状态时，依然可用。具体应用请看示例 25-3。

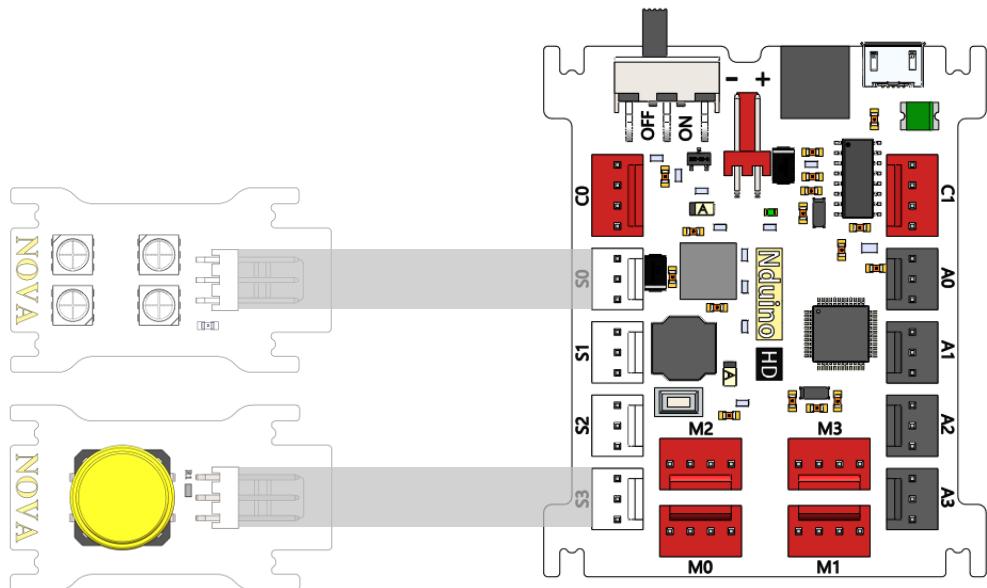
示例 23-3：按键控制切换 RGB 灯的颜色

按键按下松开一次，RGB 灯的颜色切换一次。这种应用在家里的吸顶灯中非常常见。

元器件列表：

1. Nduino HD 主控板 × 1
2. 按键模块 × 1
3. RGB 模块× 1
4. 3Pin 2510 连接线（白）× 2

电路连接：



Mixly 程序编写：

只需在示例 25-2 程序的基础上稍作修改即可：

好好搭搭硬件程序

将 mode 设定为 0

将 flag 设定为 0

重复执行

如果 读数字口 S3 = 0 且 flag = 0 那么

等待 0.05 秒

将 flag 设定为 1

如果 mode = 3 那么

将 mode 设定为 0

否则

将变量 mode 的值增加 1

如果 读数字口 S3 = 1 且 flag = 1 那么

等待 0.05 秒

将 flag 设定为 0

如果 mode = 0 那么

RGB_SET 0 0 0

如果 mode = 1 那么

RGB_SET 50 0 0

如果 mode = 2 那么

RGB_SET 0 50 0

如果 mode = 3 那么

RGB_SET 0 0 50

定义 RGB_SET R G B

RGB复位 50

发送RGB数据 50 红 R 绿 G 蓝 B

完整程序详见：<http://www.haohaodada.com/show.php?id=532165>

注意虚线红框的程序，与示例 25-2 的程序基本相同，只是 mode 的取值范围变为了 0、1、2、3。

以上程序中的输入计数用的都是按键这种数字量输入信号，如果换成模拟量信号呢？聪明的同学肯定已经想到办法了。

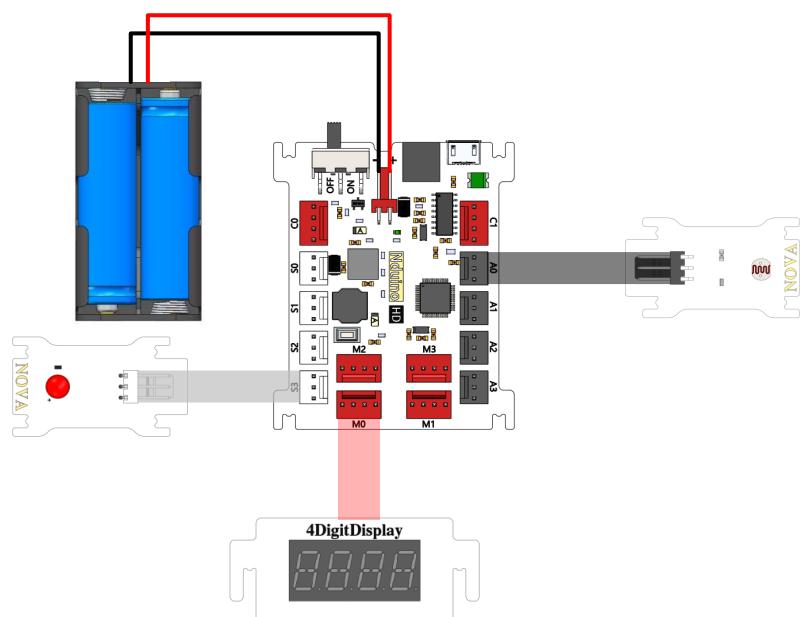
示例 23-4：亮度传感器控制 LED 灯的亮灭（点控）

摸一次亮度传感器，LED 亮，再摸一次，LED 灭，如此往复。

元器件列表:

1. Nduino HD 主控板 × 1
2. 亮度传感器模块 × 1
3. LED 模块× 1
4. 3Pin 2510 连接线（白）× 1
5. 3Pin 2510 连接线（黑）× 1

电路连接:



程序编写:

只需在示例 23-2 程序的基础上稍作修改，即将按下按钮替换为亮度值小于阈值；将松开按钮替换为亮度值大于阈值即可。所以需要先测定这个阈值。

定义一个变量 CV 为认定亮度传感器是否被触摸的阈值（Critical Value），这个阈值应该设为多少，需要做一次测定。通过串口助手打印测试亮度传感器在有无触摸时的数值。

```
1012  
1008  
1008  
1003  
1005  
1003  
1002  
1002  
1008  
1002  
1003  
1003  
1002  
1004  
1002  
1004  
1003  
1001  
1003  
1003  
1002  
1004  
1001  
1001  
1006
```

```
81  
81  
75  
80  
80  
80  
77  
80  
75  
75  
80  
78  
63  
78  
77  
79  
78  
79  
77  
73  
74  
72  
70  
72  
75
```

未触摸传感器时 触摸传感器时
阈值的选取可以从 70 到 1000 之间选取一个，本例中选取的阈值为 500。
那么程序为：



通过以上四个案例可以总结出：利用“标志位”的方法，可以让持续型信号触发一次跳变信号，避开重复运行程序导致信号多次跳变的问题。

在第十七节中，初步介绍了 MP3 模块的使用，其示例 17-1 是用红外遥控方式控制 MP3 模块，因为红外遥控信号不是持续型信号，所以重复运行程序不会造成任何影响。接下来将介绍按键、传感器这类持续型信号如何控制 MP3 模块。

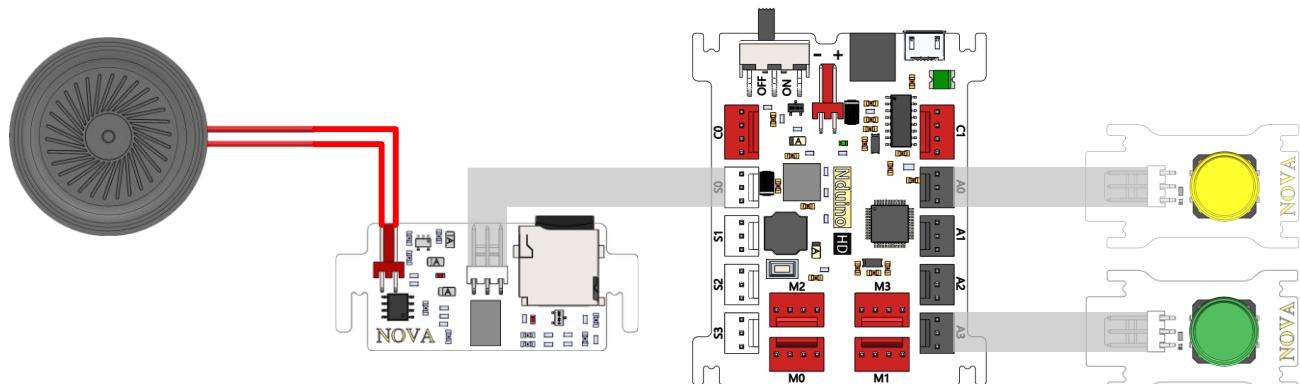
示例 23-5：用两个按键控制 MP3 的上一曲/下一曲切换

按一次按键 A，切换为上一曲；按一次按键 B，切换为下一曲。

元器件列表：

1. Nduino HD 主控板 × 1
2. 单按键模块 × 2
3. MP3 模块 × 1
4. microSD 卡 × 1
5. 扬声器 × 1
6. 3Pin 2510 连接线（白）× 3

电路连接：



程序编写：

利用 MP3 操作指令

MP3 next_song 在引脚 S0

最容易想到的程序肯定是：

好好搭搭硬件程序

重复执行

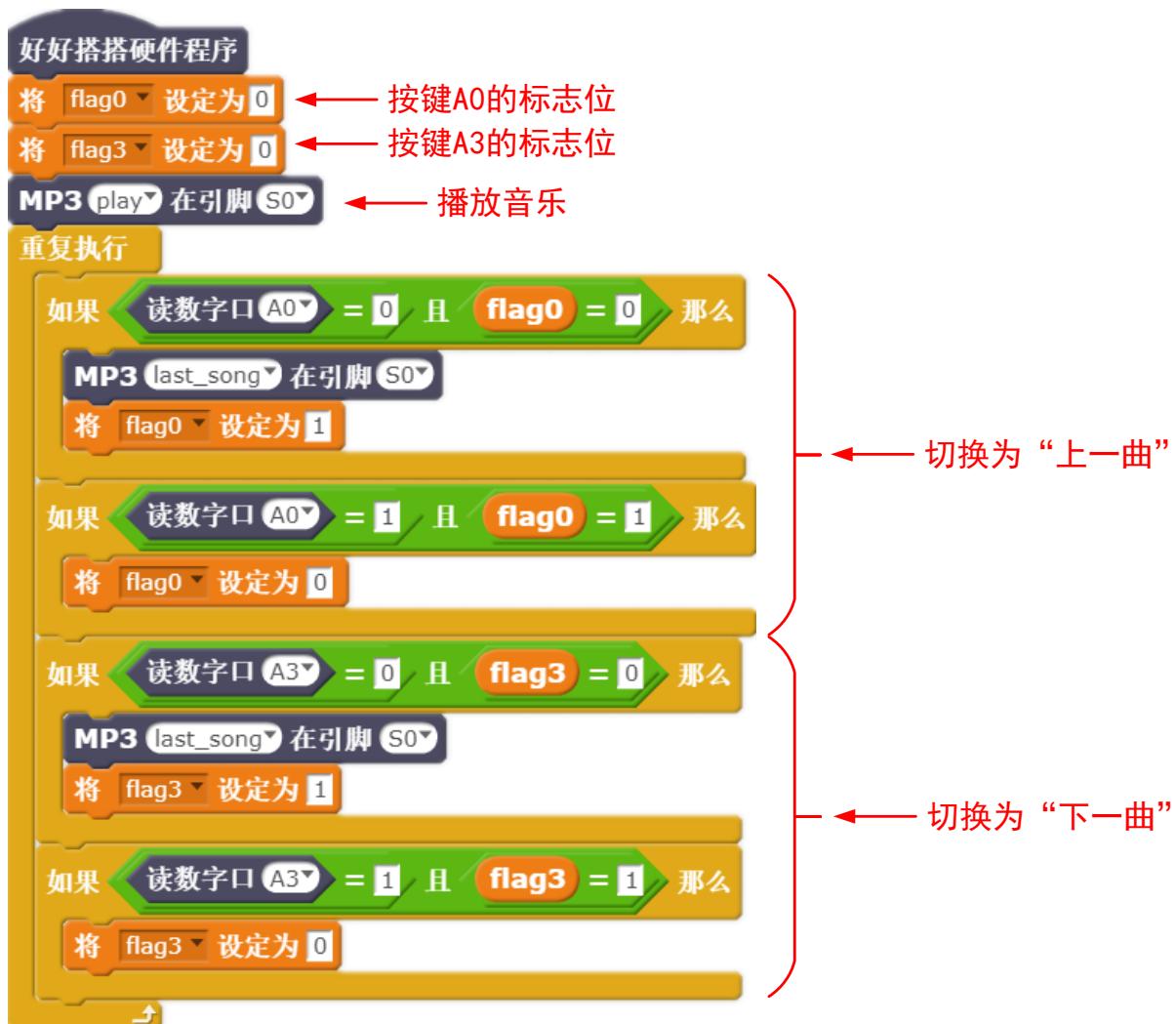
如果 读数字口 A0 = 0 那么

MP3 last_song 在引脚 S0

如果 读数字口 A3 = 0 那么

MP3 next_song 在引脚 S0

回想示例 25-1 记录按键按下松开次数的程序，就会发现以上程序的问题，按一次按键 A0，“上一曲”将会运行很多次，按一次按键 A3 同理。如果要实现按下松开按键一次，歌曲只向前或向后切换一次，那么可以仿造示例 23-1 的方式来实现。



除了“上一首”和“下一首”切换歌曲之外，MP3 模块的按曲目播放和音量加减调节程序块同样不能直接调用，反复运行。

MP3Play 1 at S0

指定曲目播放

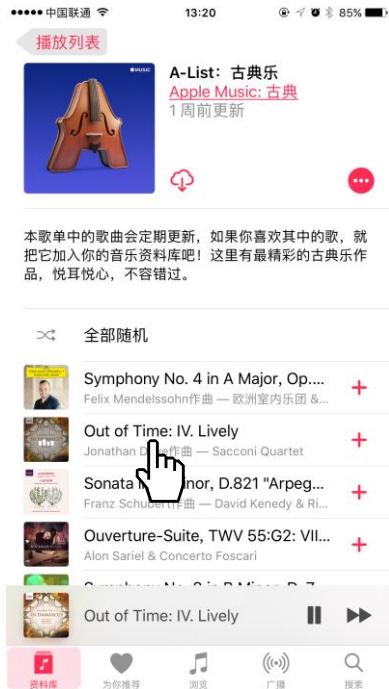
MP3 vol_up 在引脚 S0

MP3 vol_dn 在引脚 S0

音量加和音量减

音量加和音量减为什么不能直接调用，原因与上一曲/下一曲一样，会在短时间内运行很多次。

指定曲目播放为什么不能直接调用呢？我们来做一个实验，打开你手机上的音乐 APP，反复点击选取其中一首歌曲，歌曲的进度条是会反复会到歌曲的 0 分 0 秒处。正确的做法是点击选择一次，之后不再点击。



同样，使用“指定曲目播放”程序块时，也应该是运行一次之后不再运行，而不是反复运行。

第二十四节 数组

经过之前的大量案例，相信大家对变量的作用有了基本的了解。当需要很多变量，且这些变量是相关的，那么用数组将这些变量归类，并利用数组的方式存取变量值，将极大的简化程序。

数组，相当于一组带有编号的变量集合，编号本身也可以作为变量处理。

功能描述：

所谓数组，就是相同数据类型的元素按一定顺序排列的集合。

数组在程序中最大的作用是，可以把数据一个一个的放入到有编号的格子里，取用的时候也可以直接根据编号来调取。如下图 0 号格存放了数字 255、1 号格存放了数字 127 等等。

| | | | | | | |
|------|-----|-----|-----|-----|----|-----|
| 数组名 | 255 | 127 | 191 | 223 | 95 | 159 |
| 元素标号 | 0 | 1 | 2 | 3 | 4 | 5 |

数组元素的标号起始值为 0，而不是 1。

认识数组初始化指令

初始化 int 数组_① 有 ③ 项

可以通过制定数据类型定义数组元素的数据类型，其中

- (1) int 为有符号整型，取值范围为-2147483648~2147483647;
- (2) uint32_t 为无符号 32 位整型，取值范围为 0~4294967295;
- (3) uint8_t 为无符号 8 位整型，取值范围为 0~255;
- (4) char 为字符型;
- (5) float 为浮点型，即带小数点的数字。

如上图指令初始化的数组，有三个元素，标号分别为 0、1、2，每个元素的值均为 0。

认识数组元素赋值指令

设置数组_① 第 ② 项为 ③

标号和数值都可以用变量来替换



认识数组元素读取指令

读取数组_ 1 第 0 项

可以读取某个数组元素的值。

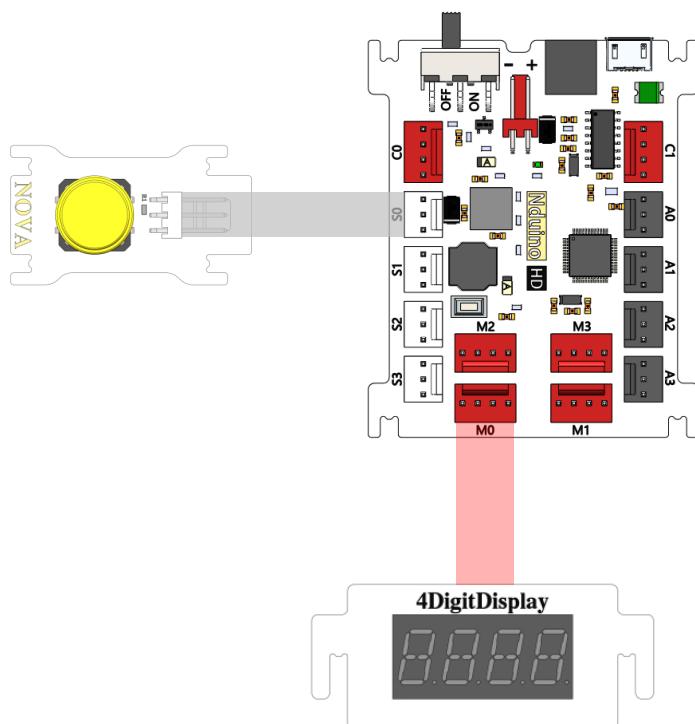
示例 24-1：抽签机随机取数不重复

按下松开一次按键，数码管显示一个数字，每次显示的数字均不重复。显示完毕之后数码管显示 8888。

元器件列表：

1. Nduino HD 主控板 ×1
2. 单按键模块 ×1
3. 数码管模块 ×1
4. 3Pin 2510 连接线（白）×1
5. 4Pin 2510 连接线（红）×1

电路连接：



程序编写:

方案 1: 检查法

方法概述:

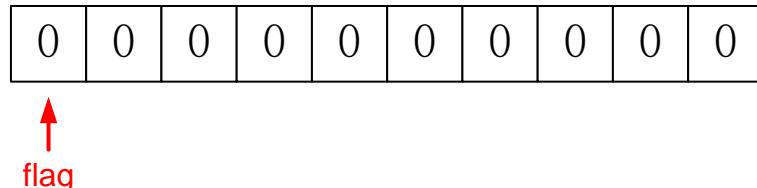


程序分析:

第一步, 建立一个 10 位的空数组:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|

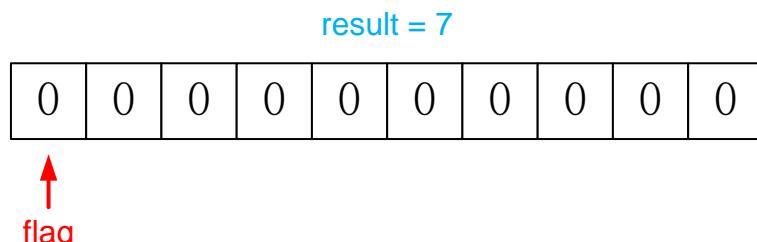
第二步, 定义一个变量 flag, 用来指向数组的一个元素, 初始时指向标号 0 的元素:



第三步: 按下按钮, 只要不松开, 不断的取出随机数, 将结果暂时存放在一个变量 result 里, 将变量 result 去与数组中从标号 0 元素到标号 flag-1 的元素一一比较。

推演: 当 flag=0 时, 标号 0 元素的值为 0, result 的值为 1~10 中的一个, 必然不相等,

将 result 的值存入数组标号 0 元素中, 假设其值为 7。



flag 加 1, 指向后一位, 为下一次抽取做准备。

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | |

↑
flag

此时 $\text{flag}=1$ 。按下按键， result 继续从 1~10 中随机取出一个，如果 result 取中的仍然是 7，则这个结果是不符合要求的，必须重新抽取。问题是如何判断结果是否符合要求？

$\text{result} = 7$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | |

↑
flag

当 $\text{flag}=1$ 时， result 要与标号 0 元素进行比较，如果相等，则不符合要求，继续抽取；当 $\text{flag}=6$ 时， result 要与标号 0 元素到标号 5 元素一一比较。推而广之，即当 $\text{flag}=n$ 时， result 需要跟数组前 n 个元素都进行比较，确认 result 与它们都不同，才是符合要求的结果。

$\text{result} = 9$

| | | | | | | | | | |
|---|---|---|----|---|---|---|---|---|---|
| 7 | 3 | 5 | 10 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | |

↑
flag

$\text{result} = 5$

| | | | | | | | | | |
|---|---|---|----|---|---|---|---|---|---|
| 7 | 3 | 5 | 10 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | |

↑
flag

✓

✗

可以用一个循环程序块，让 result 去与标号 0 的元素比较，一直比较到标号 $\text{flag}-1$ 元素。

定义一个变量 check ，初始值为 0， result 每比较一个元素，如果不相等，则 check 的值加 1，如果相等，则 check 的值不变。那么如果 result 的值与已抽出数字全部不同时，则 check 的值等于 $\text{flag}+1$ ，如果 result 的值与已抽出数字有相同时，则 check 的值必然不等于 $\text{flag}+1$ 。所以 check 的结果可以用来判断抽出的 result 是否符合要求。

最后，当 10 个数字全部抽取出来之后，数码管可以显示 8888，已提醒用户抽签结束。

图形化程序如下：

初始化，定义相关变量和数组



主程序：



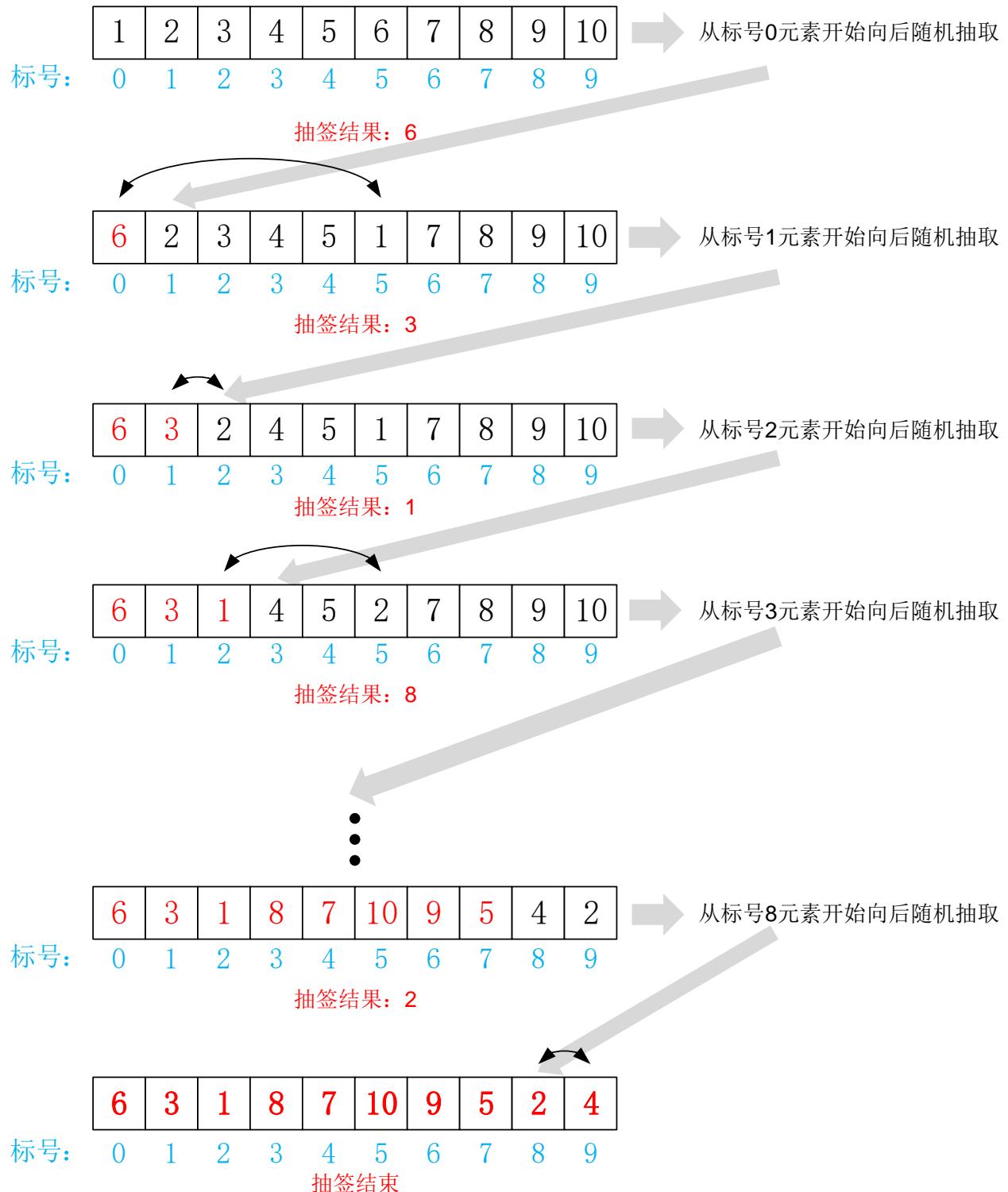
完整程序详见：<http://www.haohaodada.com/show.php?id=534466>

项目要求二——方案 2：交换法

方法概述：

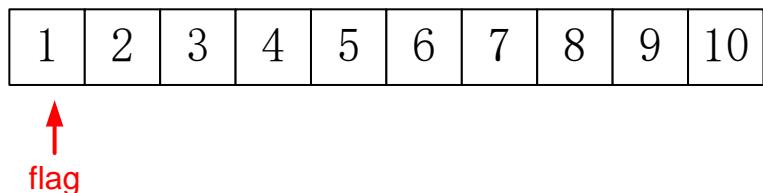
先在数组按顺序存入 1~10 十个数字；

再不停的抽取数字，并进行调换。举例说明：

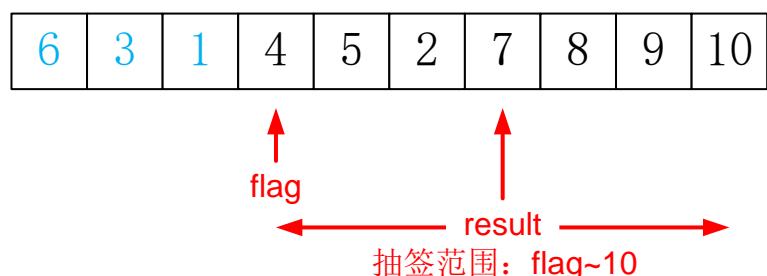


程序分析：

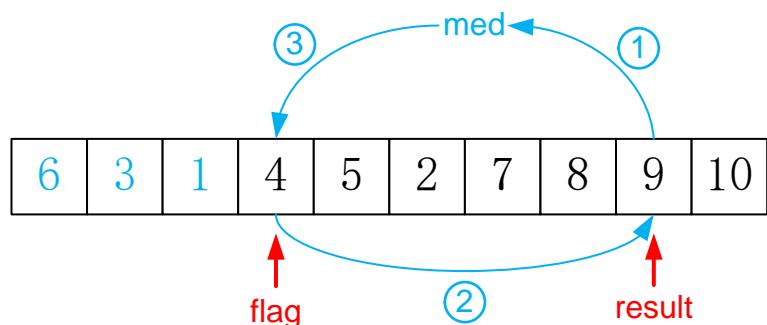
第一步，建立一个 10 位数组，把 1~10 十个数字按顺序存放进去。同方法 1 一样，定义一个数组标号变量 flag，初始值为 0。



第二步：按下按钮，只要不松开，不断的从第 flag 元素到第 10 个元素中取出随机数，存入变量 result，并把数组中标号为 result 的元素显示在数码管上。



第三步：交换数值，定义一个中间变量 med 来帮助实现 flag 和 result 指向的元素进行交换。



第四步：flag 的值加 1，进行下一次的抽取。

第五步：全部抽取结束，flag 的值为 11，再按一下按钮，数码管显示 8888。

图形化程序如下：

初始化，定义相关变量和数组



主程序：



第五部分

机器人应用

第二十五节 智能小车的运动

在介绍智能小车遥控、避障、循线之前，有必要介绍智能小车的运动时的电机参数。本书以三轮小车为例展开，三轮小车由两个主动轮和一个万向轮组成。智能小车的前进、后退、左转和右转即是通过两个主动轮的不同速度组合实现的。

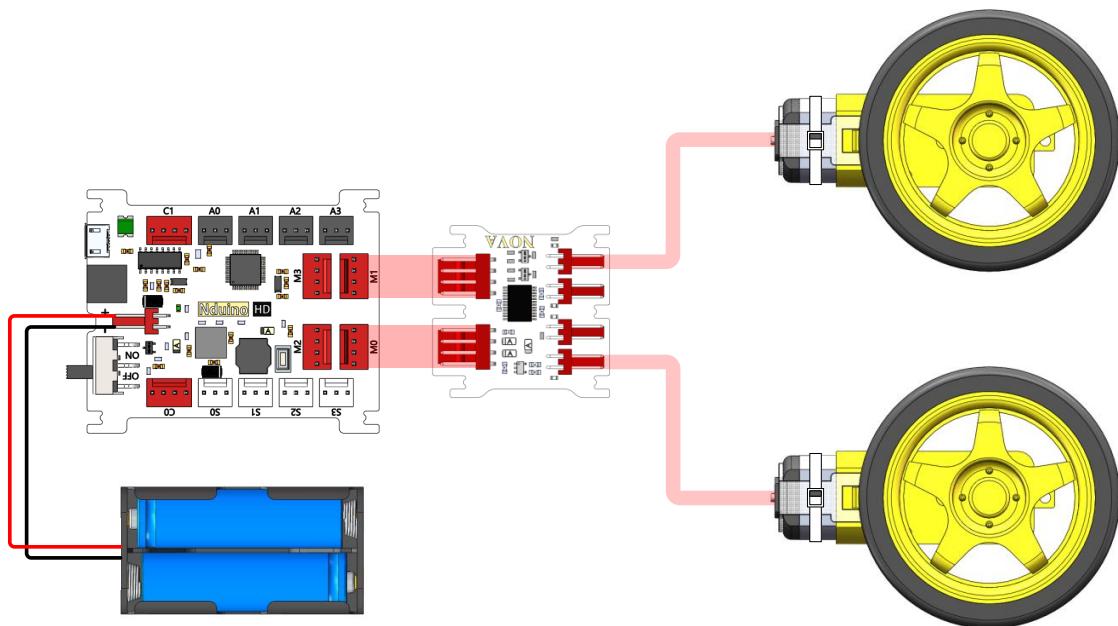
示例 25-1：智能小车的前进和后退

打开开关，小车一直前进或后退

元器件列表：

1. NOVA 智能小车（含主控、超声波、双电机驱动模块、灰度传感器） ×1

电路连接：



程序编写：

利用双电机驱动指令，给两个电机分别写入速度值。

好好搭搭硬件程序

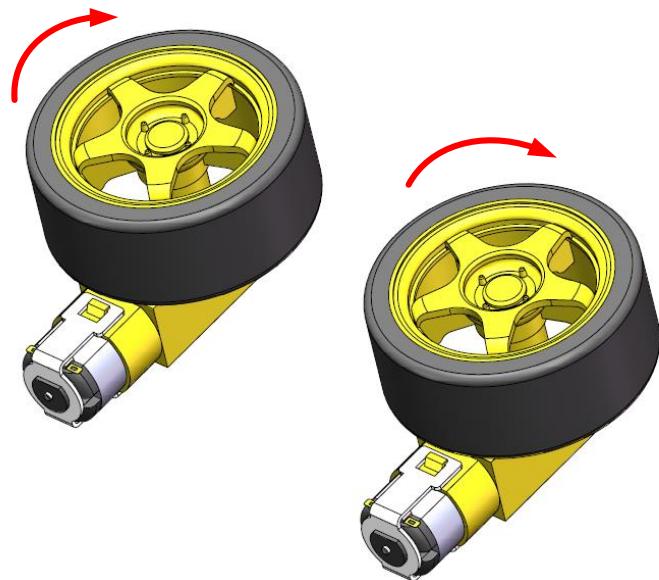
双电机驱动 M0 电机输出 150

双电机驱动 M1 电机输出 150

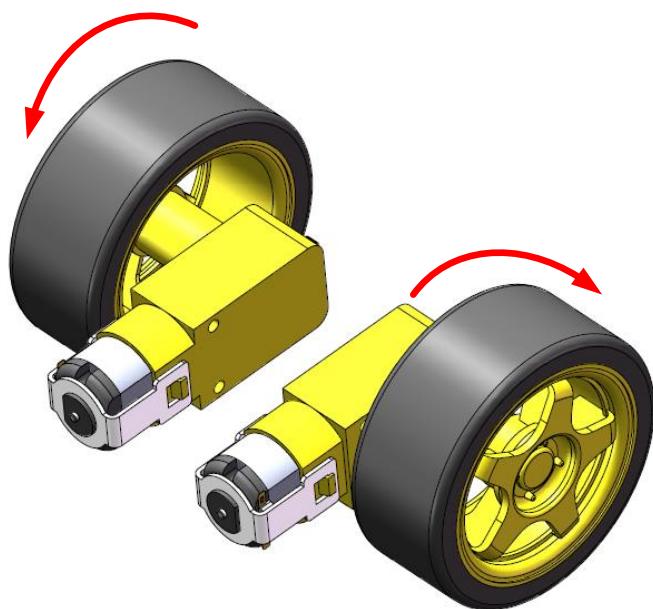
两个电机都写入相同的数值 150，程序运行之后会发现小车并不向前走，而是原地转圈。

分析原因：

- (1) 当两个电机朝向相同摆放时，可以看出，两轮均按顺时针旋转；



(2) 当两个电机朝向相反时，虽然轮子相对电机仍然是顺时针，但是相对小车来说，一个向前提供驱动力，一个向后提供驱动力，导致小车原地转圈。



所以，让小车前进的程序，应为

好好搭搭硬件程序

双电机驱动 M0 电机输出 150

双电机驱动 M1 电机输出 -150

后退的程序，同学们自己试试看。

示例 25-2：智能小车的左转和右转

打开开关，小车一直左转或右转

元器件列表：

- NOVA 智能小车（含主控、超声波、双电机驱动模块） ×1

电路连接：与示例 25-1 相同

程序编写：

在示例 25-1 中，小车前进的参数为左轮（M1）速度值为-150，右轮（M0）速度值为 150。

这里将小车左轮（M1）的速度值改为-50，右轮（M0）速度值依然为 150。



程序运行结果右轮速度高，左轮速度低，左转成功。

同学们再试试以下几种速度组合：

- 左轮（M1）的速度值为 0，右轮（M0）速度值为 150；
- 左轮（M1）的速度值为 50，右轮（M0）速度值为 150；
- 左轮（M1）的速度值为 150，右轮（M0）速度值为 150；
- 左轮（M1）的速度值为 100，右轮（M0）速度值为 200。

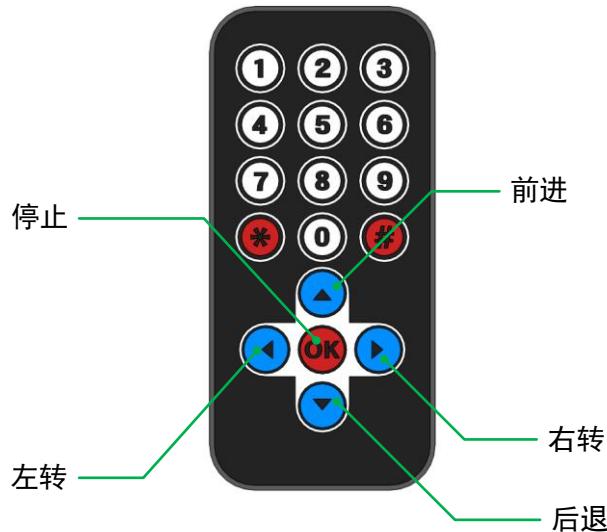
根据以上的速度组合的运行效果，同学们试分析小车左右轮参数与小车转弯半径的关系。

第二十六节 红外遥控智能小车

将红外遥控与电机驱动结合起来，就能实现通过红外遥控器远程控制小车运动。

示例 26-1：红外遥控智能小车

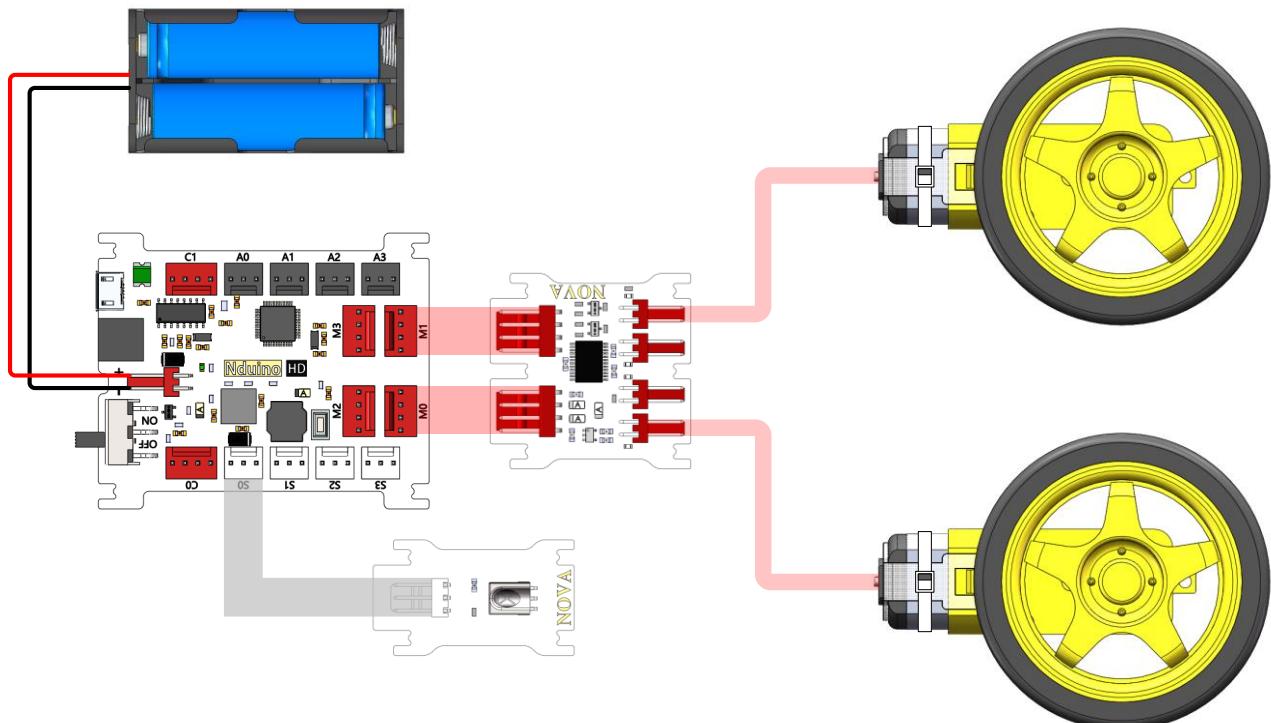
通过红外遥控器控制小车的运动，设定如下：



元器件列表：

- NOVA 智能小车（含主控、超声波、双电机驱动模块、灰度传感器） ×1

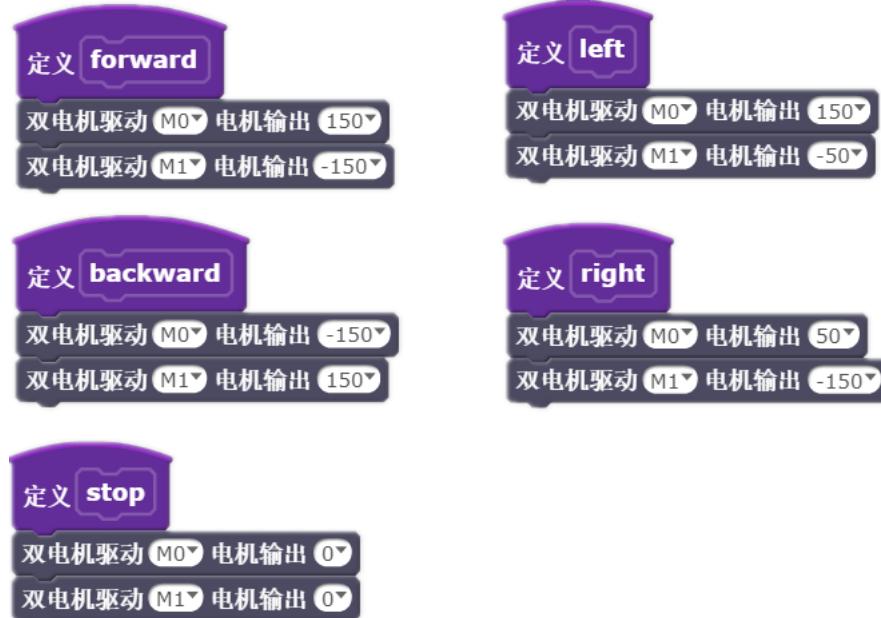
电路连接：



程序编写：

在第二十五节中，同学们已经掌握了小车前进、后退、左转、右转、停止的编程方法，但是这些程序段都是由指令和参数组成的，很难被记忆和阅读。借用第十五节讲到的“新建功能块”的方法能有效的提升程序的可读性。

先新建小车前进、后退、左转、右转、停止五个功能块。



再编写红外遥控的主程序



第二十七节 智能小车实现自动避障

自动避障是无人驾驶汽车的基本功能，这节课的内容是介绍如何利用超声波测距传感器实现智能小车的自动避障。

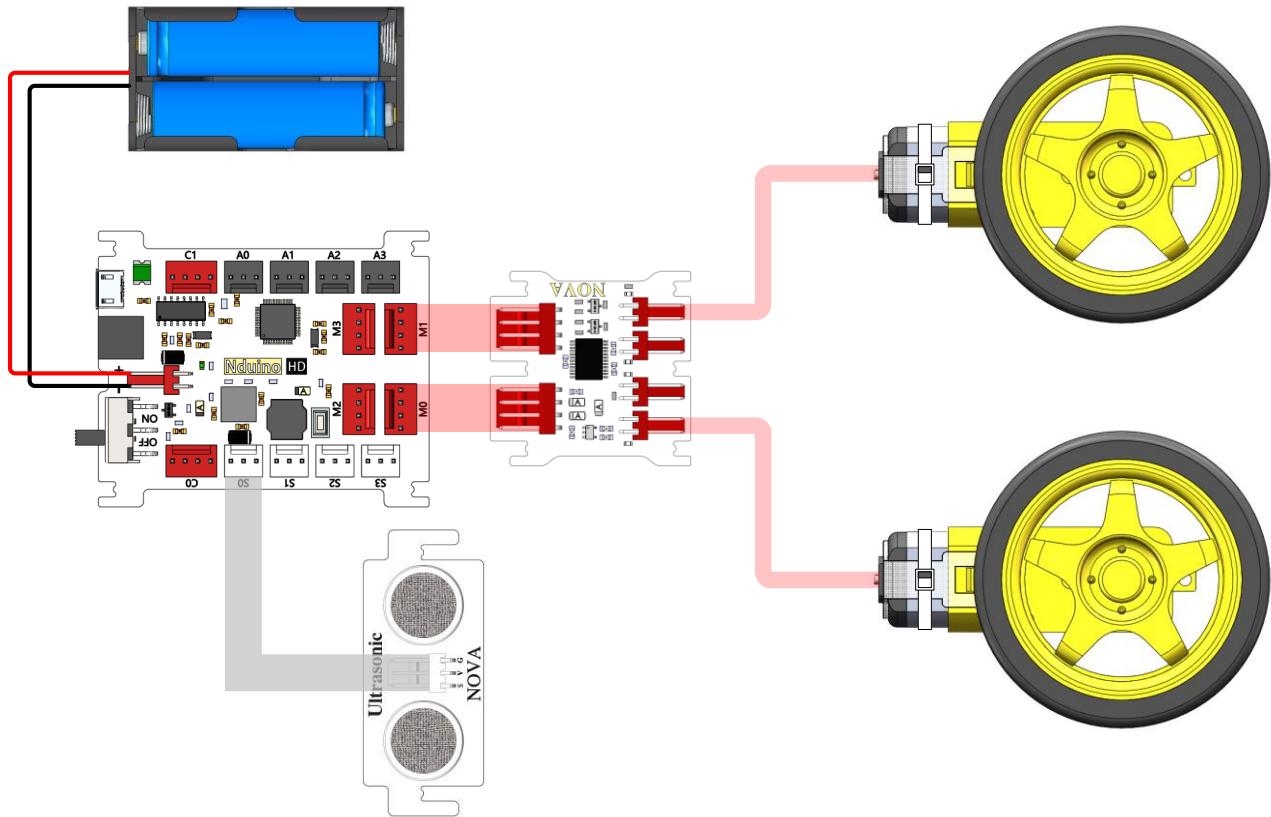
示例 27-1：自动避障智能小车

当小车前方 15cm 范围内有障碍物时，小车转动改变行进方向，无障碍物时，则小车一直向前行进。

元器件列表：

1. NOVA 智能小车（含主控、超声波、双电机驱动模块、灰度传感器） ×1

电路连接：



程序编写：

在第二十六节中新建好的前进、后退、左转、右转、停止功能块，将在本示例中继续使用。

好好搭搭硬件程序

重复执行

如果 读2.0超声波传感器在 S0 > 15 那么

forward

前方15cm范围内没有障碍物，前进

如果 读2.0超声波传感器在 S0 < 15 那么

将 a 设定为 在 1 到 2 间随机选一个数

前方15cm范围内有障碍物，进入避障程序

如果 a = 1 那么

backward

等待 0.5 秒

right

等待 0.5 秒

在转弯之前，先后退一段距离

如果 a = 2 那么

backward

等待 0.5 秒

left

等待 0.5 秒

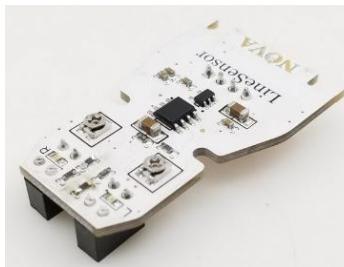
随机选择左转还是右转

第二十八节 自动循线智能小车

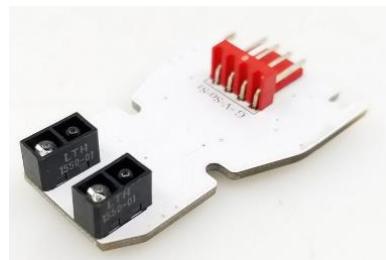
机器人的自动循线行进在工业中被广泛的应用，是目前最低成本最可靠的机器人运动方式。这节课的内容是介绍如何利用灰度传感器实现智能小车的自动循线。

认识灰度传感器

用来检测目标物体的黑白度，集成两个灰度传感器探头。



正面



背面

认识读灰度传感器指令

读巡线传感器 L_line 在 C0

通过该指令读取灰度传感器的值。通过选择 L_line 和 R_line 来分别读取左右两个探头的检测值。

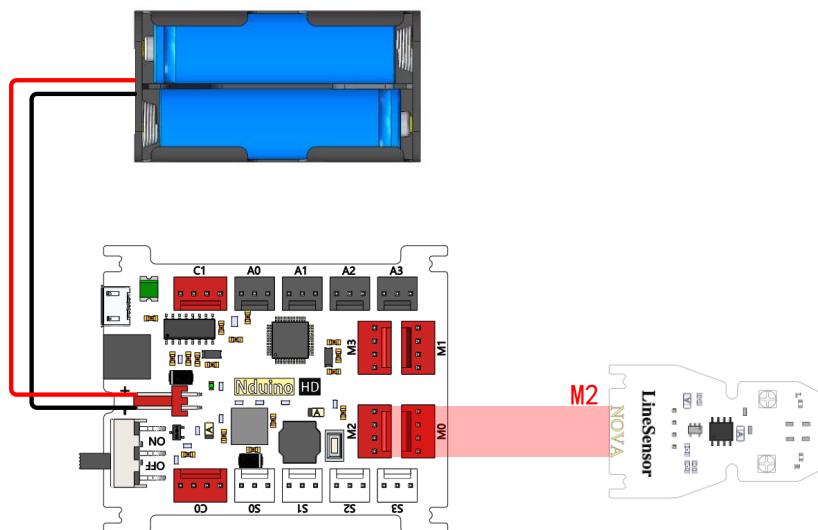
示例 28-1：将灰度传感器的值显示到串口助手中去

将灰度传感器左右两个探头的值同时显示到串口助手中

元器件列表：

1. NOVA 智能小车（含主控、超声波、双电机驱动模块、灰度传感器） ×1

电路连接：



调试传感器:

注意：请将灰度传感器安装在小车结构件上调节！以下图示是为了完整的展示操作步骤和细节。

使用螺丝刀旋转背面的电位器可以调节识别黑白线的阈值。

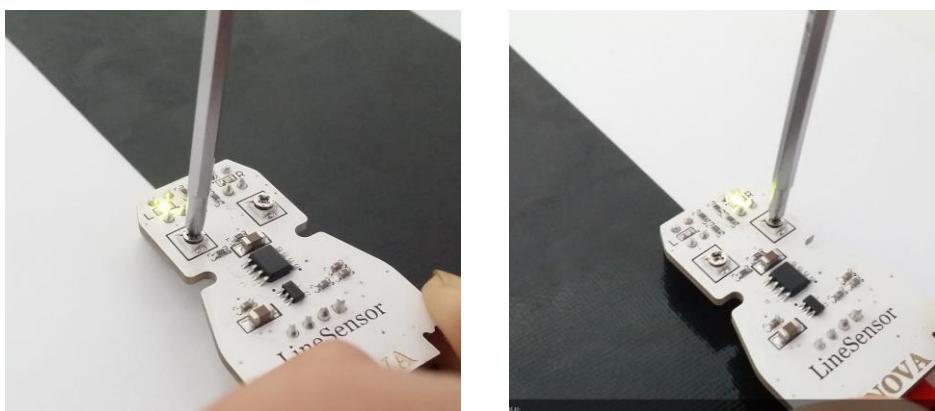
背面两个 LED 指示灯分别用于指示传感器识别状态，当 LED 灯熄灭时，代表传感器对准黑色，当 LED 灯亮起时，代表传感器对准白色。

调节步骤：

将两个灰度传感器都对准黑色块，调节背面电位器，以使两个 LED 指示灯都熄灭。



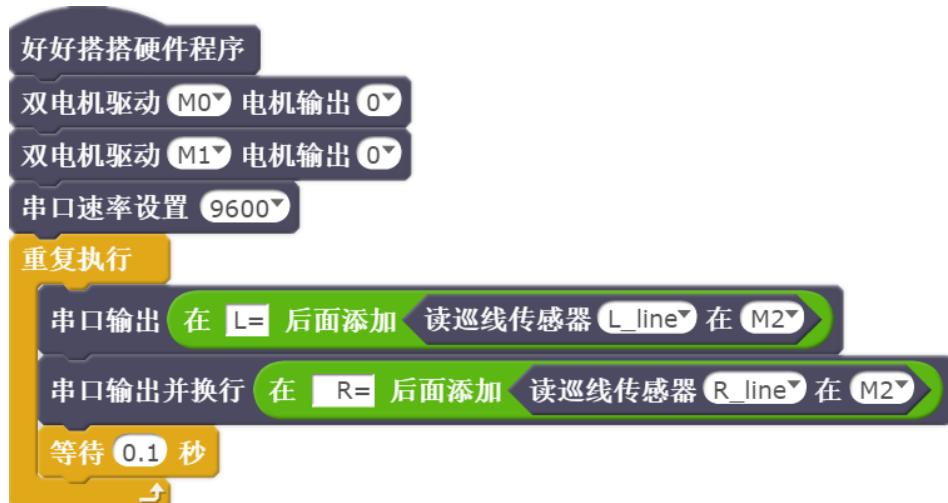
将左侧灰度传感器对准白色快，右侧灰度传感器对准黑色块，调节背面电位器，以使左侧 LED 指示灯（L）亮起，右侧 LED 指示灯（R）熄灭。反之同理



最后，将两个灰度传感器都对准白色块，两个 LED 灯都会亮起。



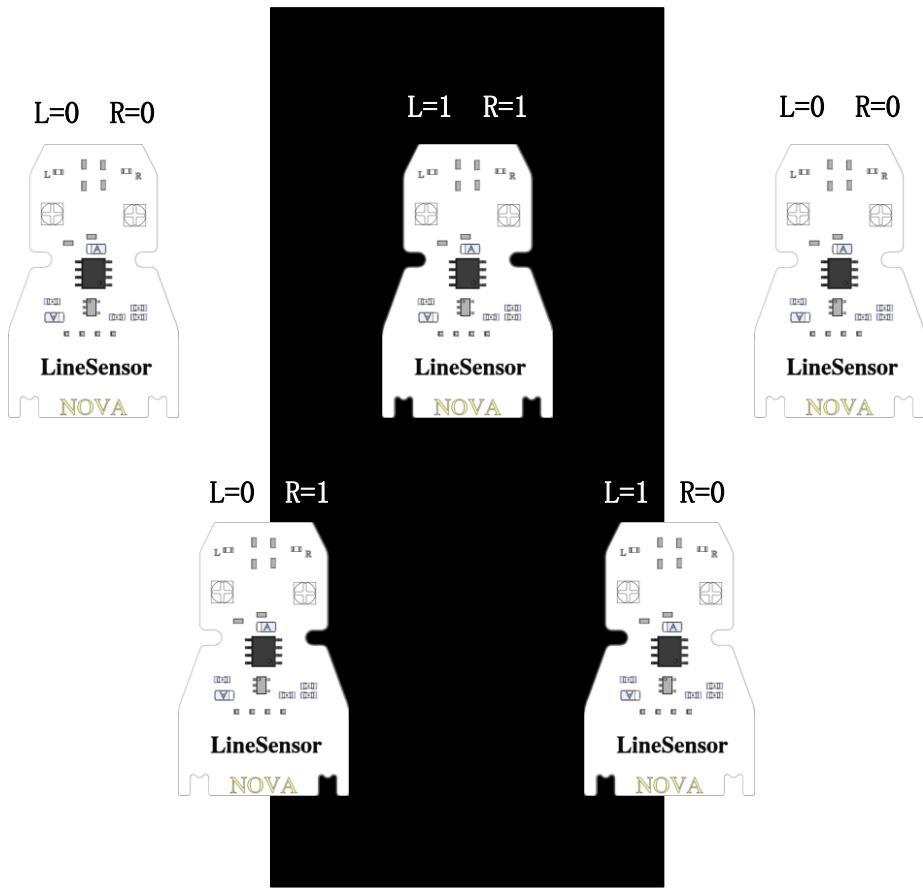
程序编写:



程序运行结果：



以上程序的显示结果，可以总结出以下结果。



示例 28-2：智能小车的自动循线

让智能小车沿 8 字地图轨迹自动循线行进。

元器件列表：

1. NOVA 智能小车（含主控、超声波、双电机驱动模块、灰度传感器） ×1

电路连接：与示例 28-1 相同

程序编写：

智能小车的自动循线也是根据灰度传感器的返回值做出相应判断的。

- (1) 如果左右两个探头都检测到的是黑色，则小车直线前进；
- (2) 如果左探头检测到的白色，右探头检测到的黑色，说明小车向左偏离，则右转；
- (3) 如果左探头检测到的黑色，右探头检测到的白色，说明小车向右偏离，则左转；
- (4) 如果左右两个探头都检测到的是白色，说明小车前冲出界，则小车直线后退；

所以程序为：

好好搭搭硬件程序

重复执行

如果 读巡线传感器 L_line 在 M2 = 1 且 读巡线传感器 R_line 在 M2 = 1 那么

forward

如果 读巡线传感器 L_line 在 M2 = 1 且 读巡线传感器 R_line 在 M2 = 0 那么

left

如果 读巡线传感器 L_line 在 M2 = 0 且 读巡线传感器 R_line 在 M2 = 1 那么

right

如果 读巡线传感器 L_line 在 M2 = 0 且 读巡线传感器 R_line 在 M2 = 0 那么

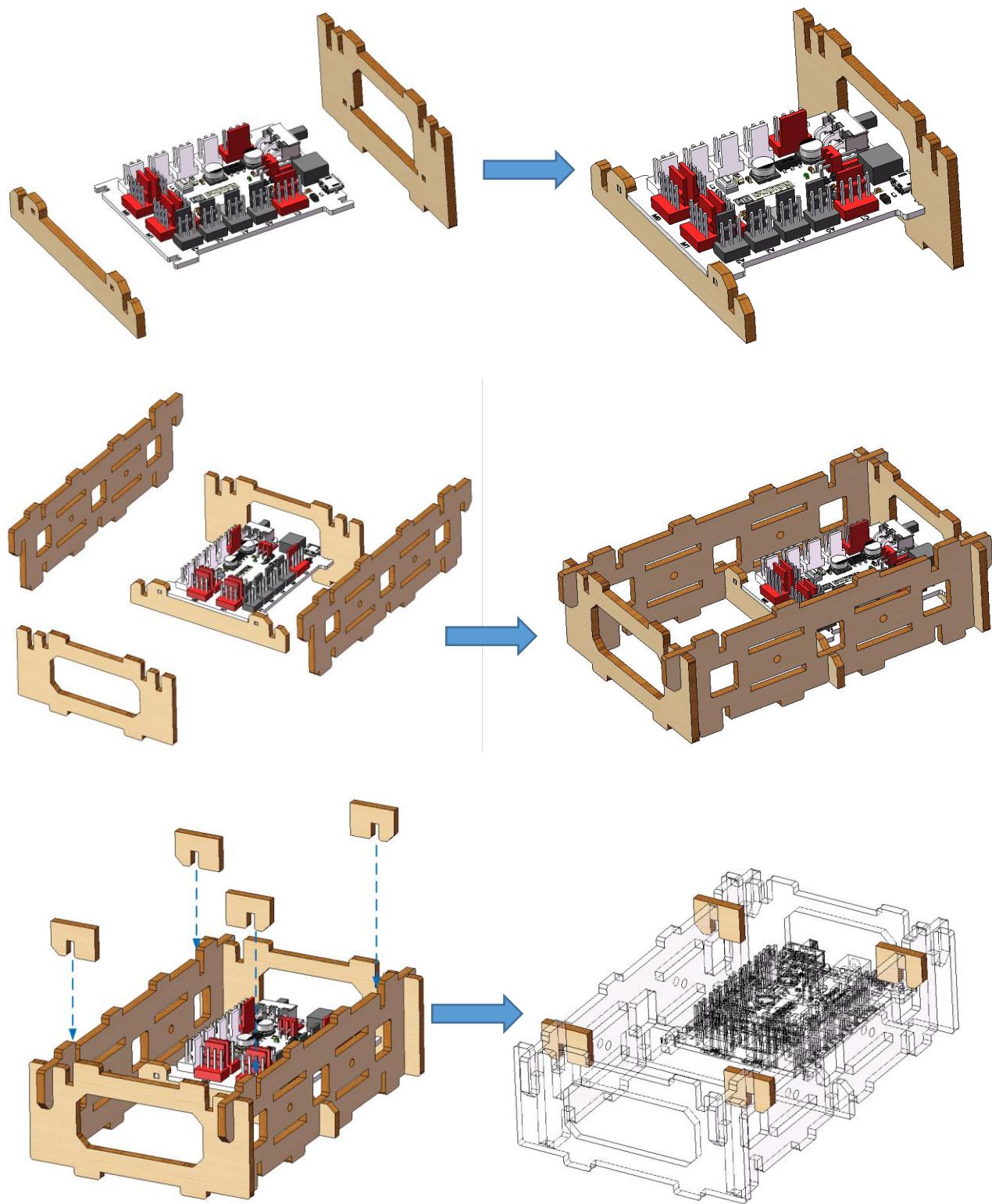
backward

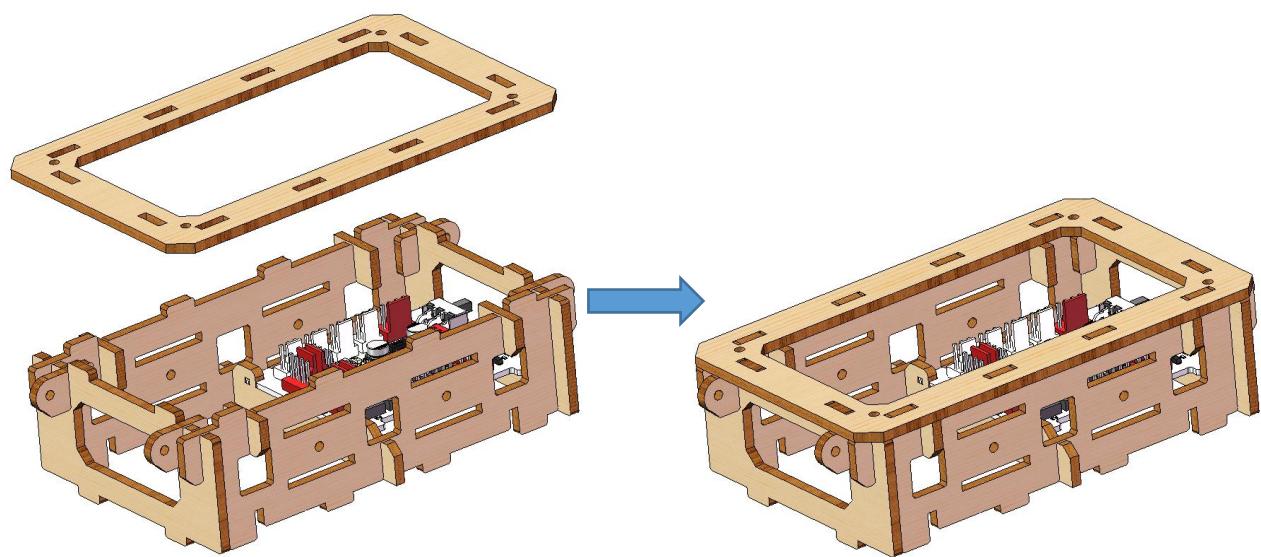
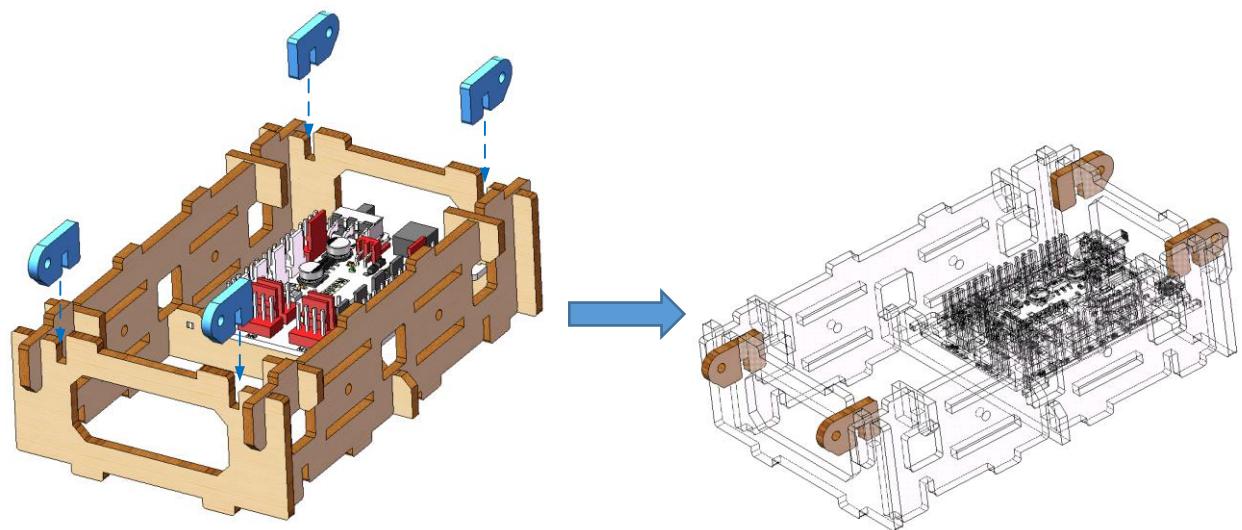


以上是智能小车自动循线程序的基本框架。为了让小车的行进效果尽量流畅，则需不断调试前进、左转和右转的速度参数。

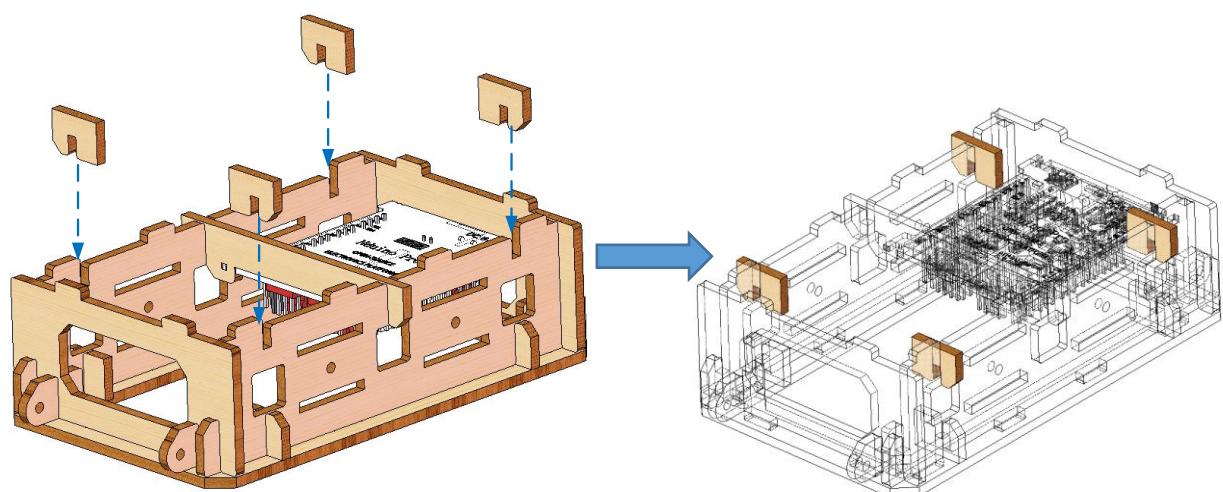
附录一 百变小魔盒拼装说明书

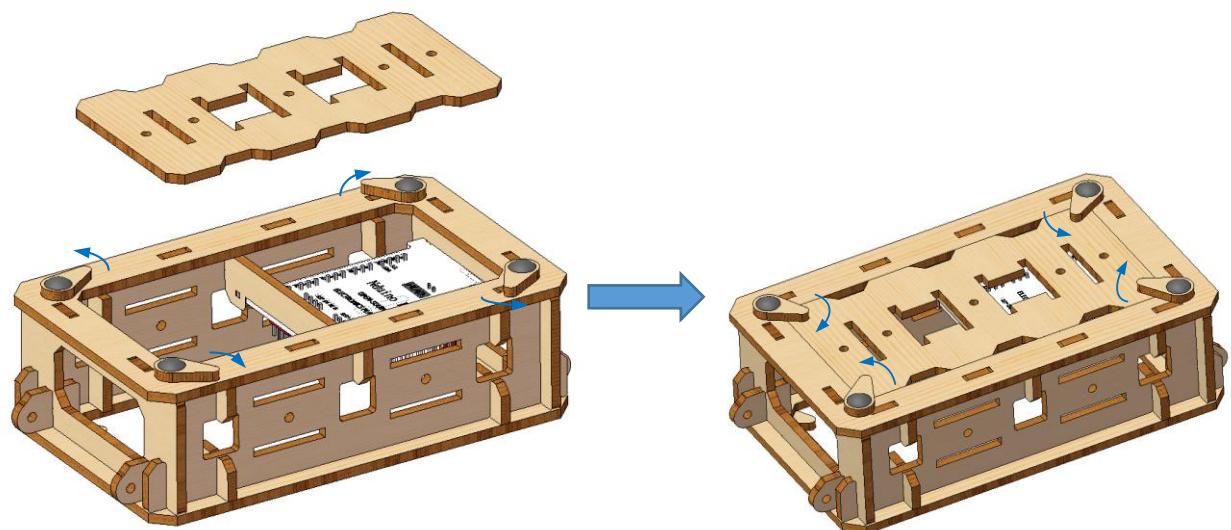
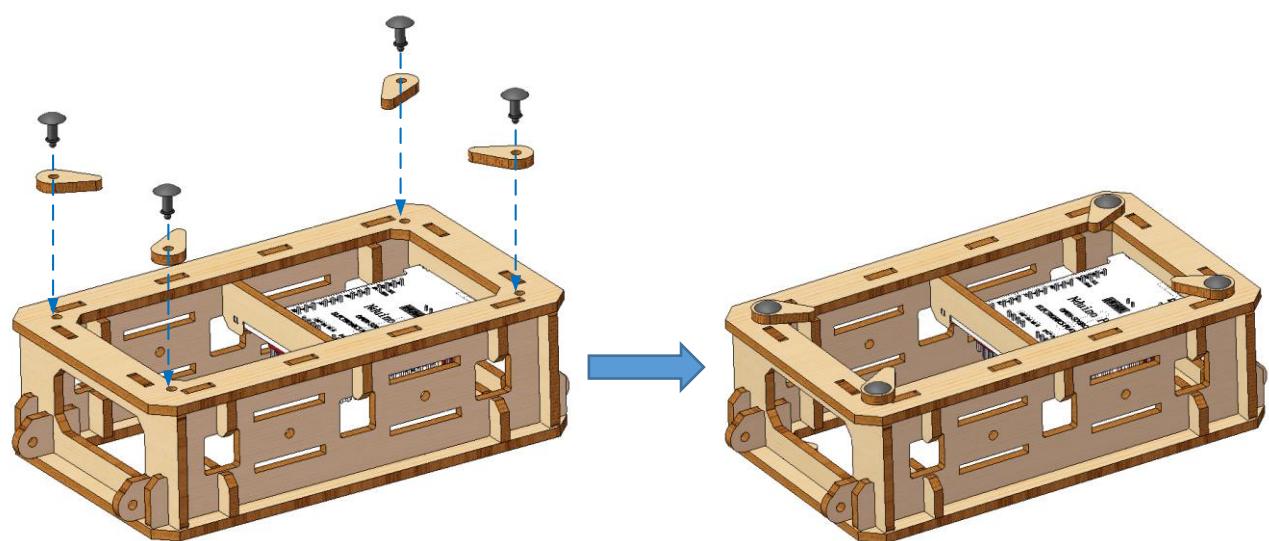
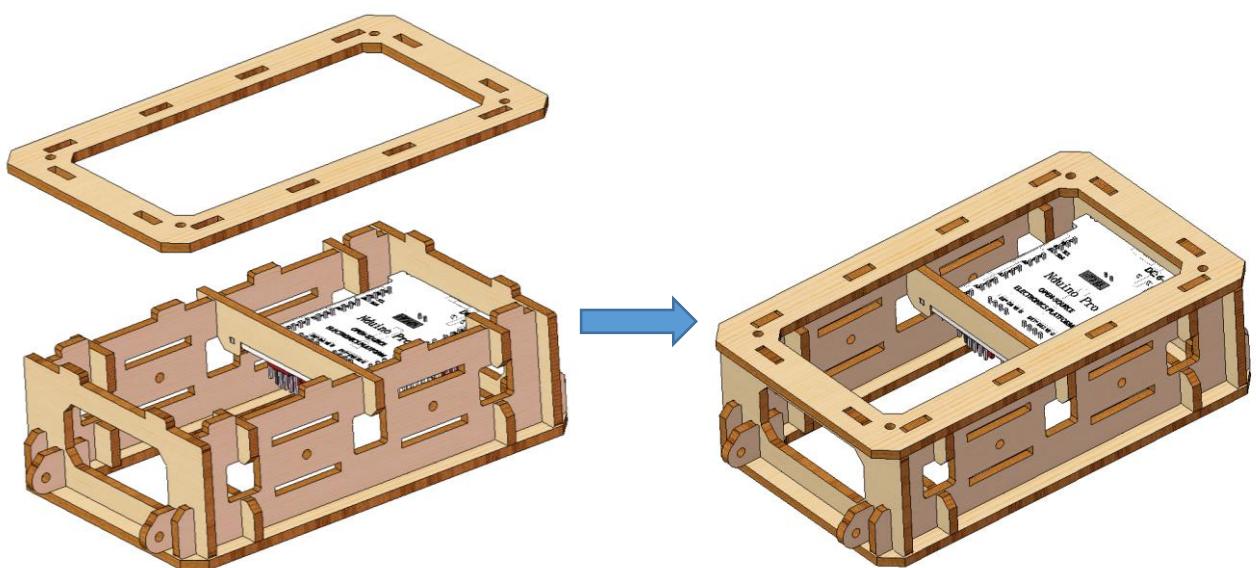
(1) 安装基础框架

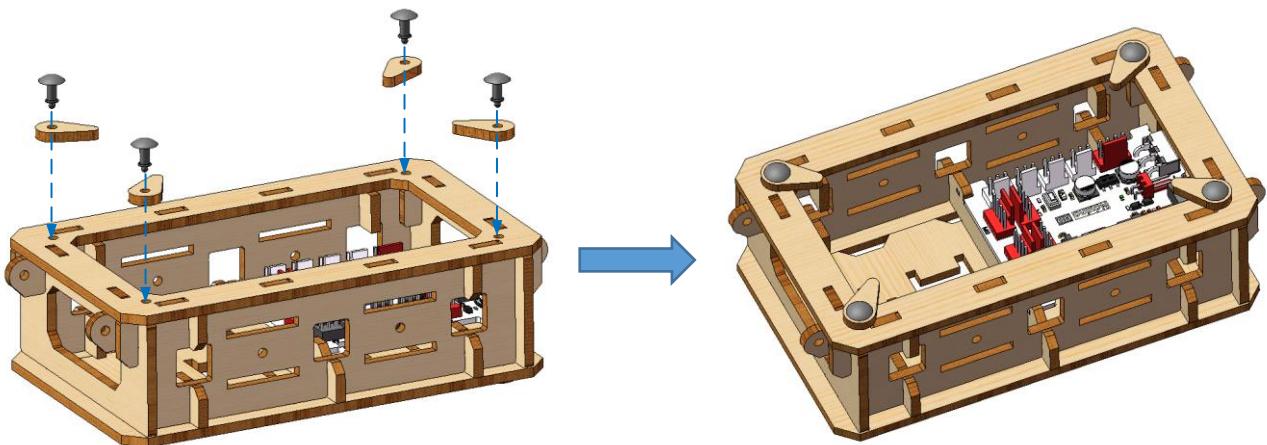




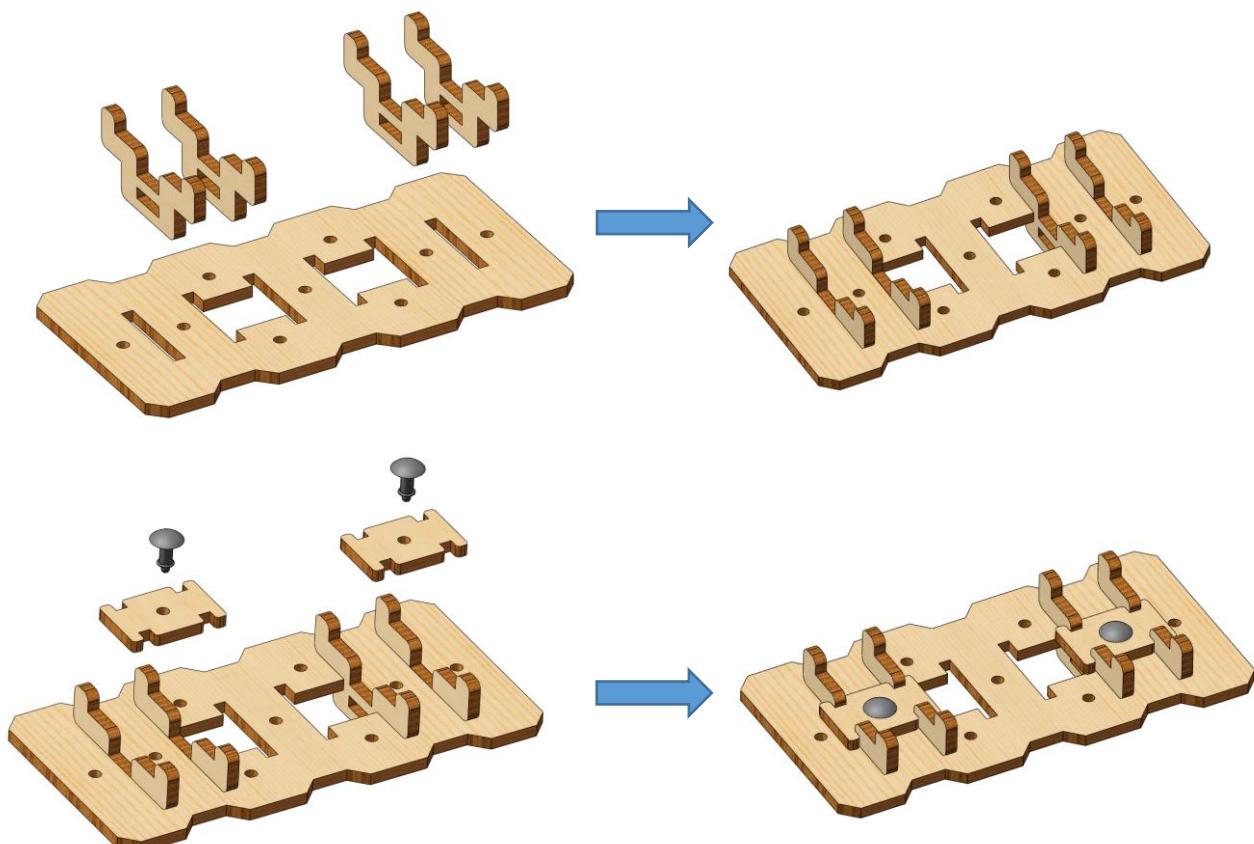
翻转

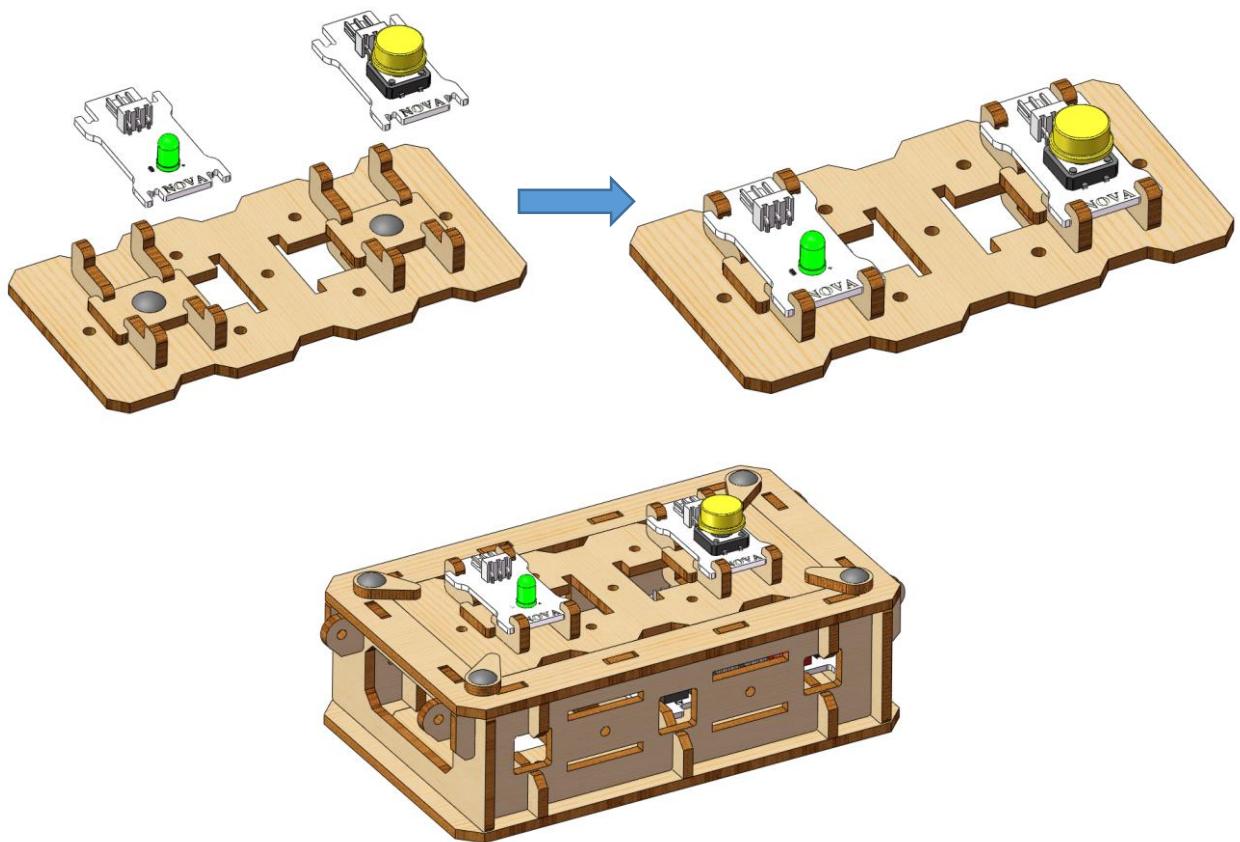




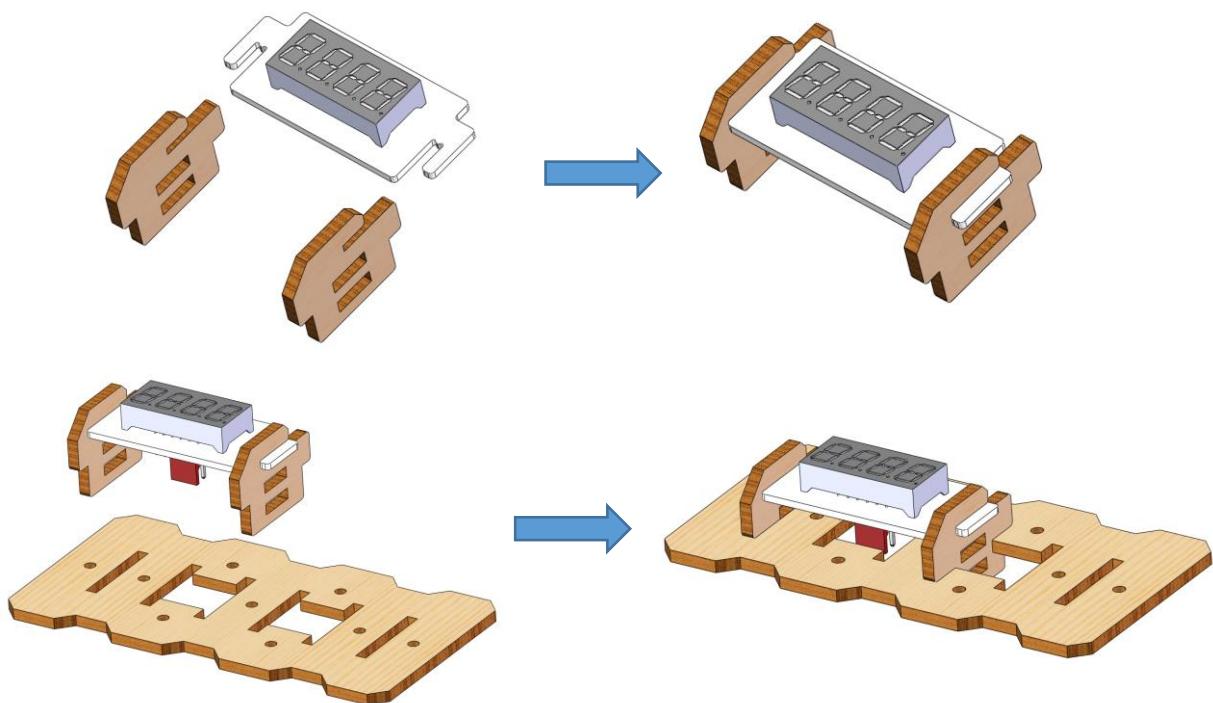


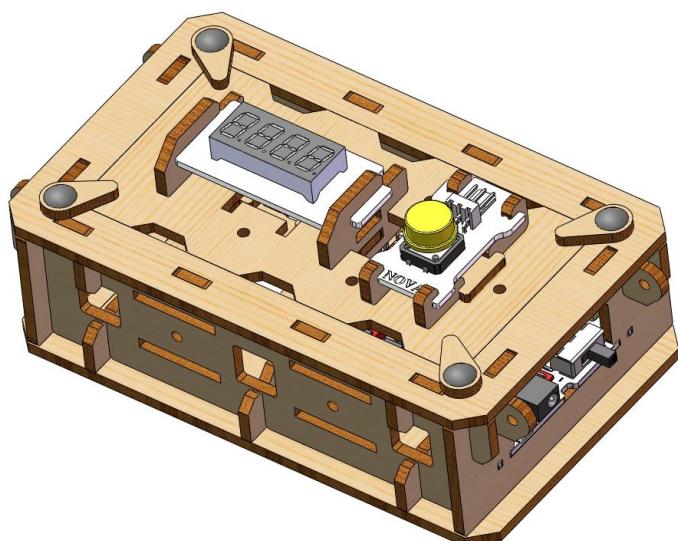
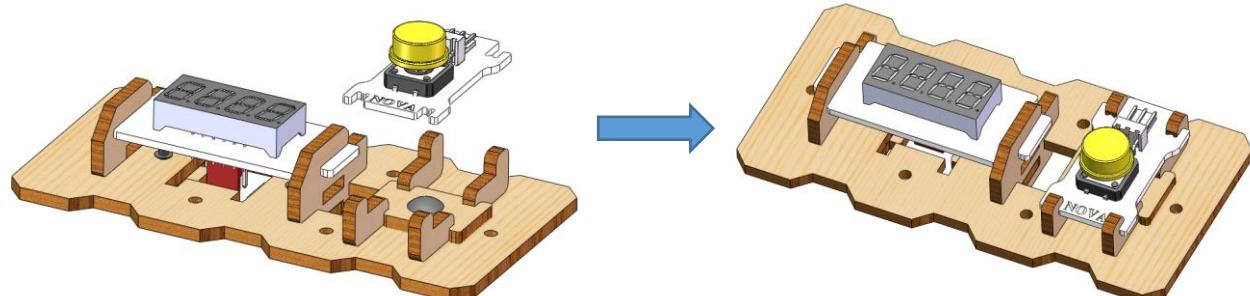
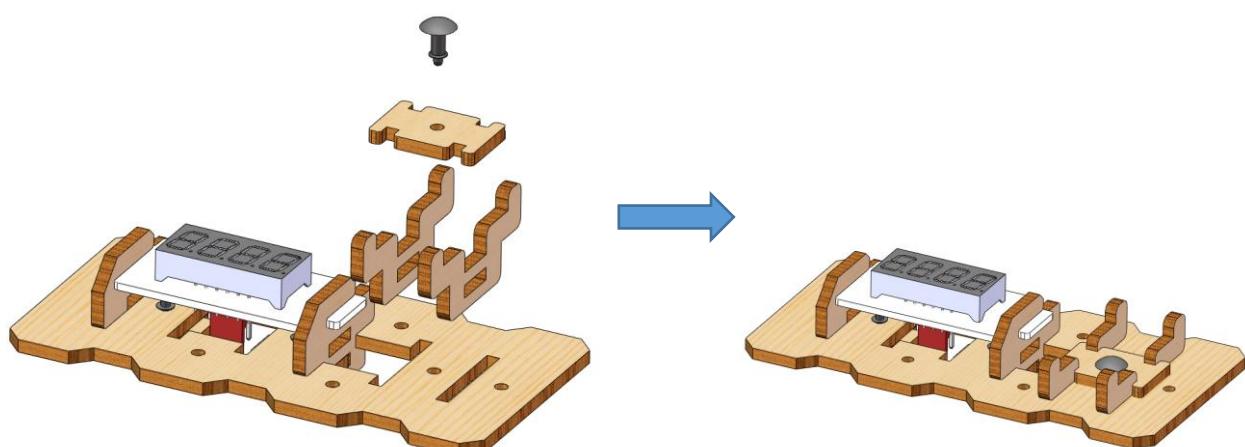
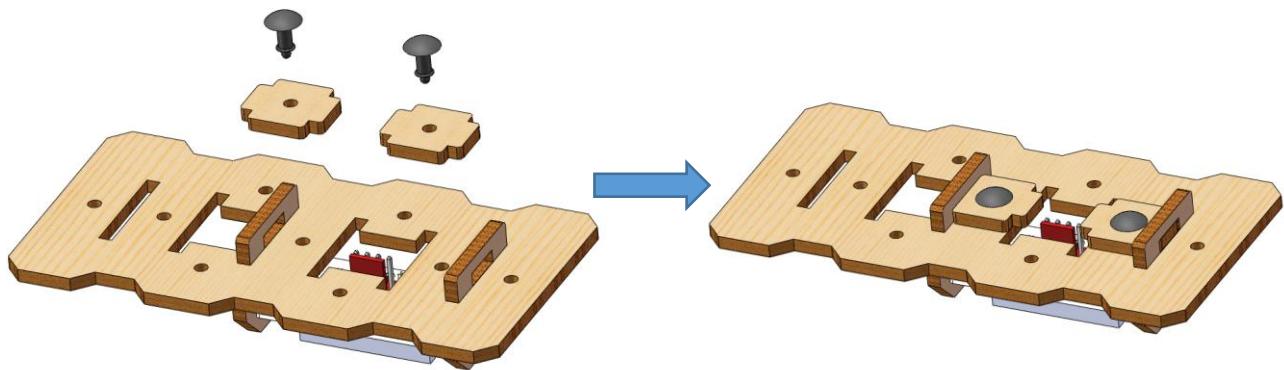
(2) 结构案例
按键控制 LED 灯



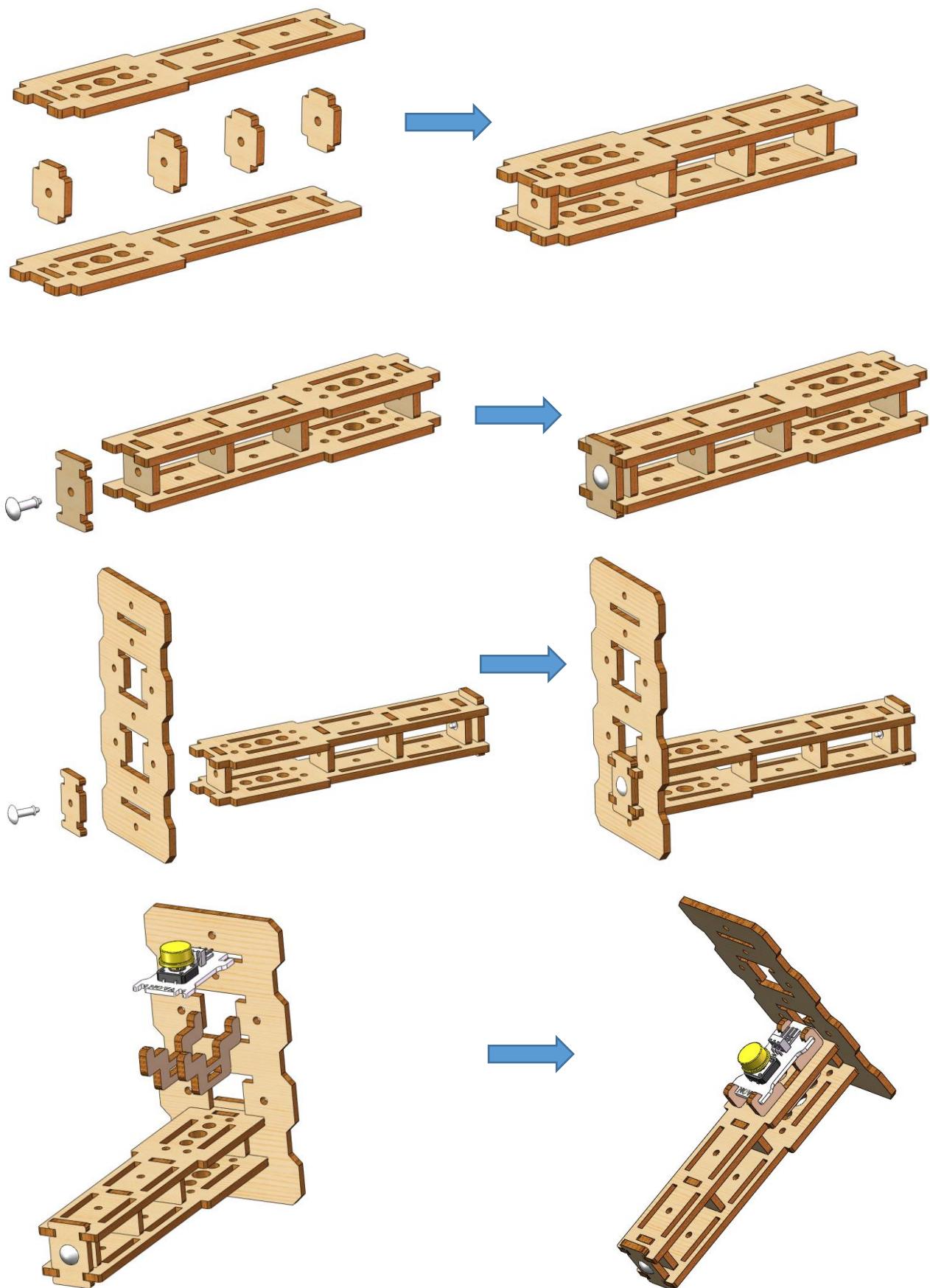


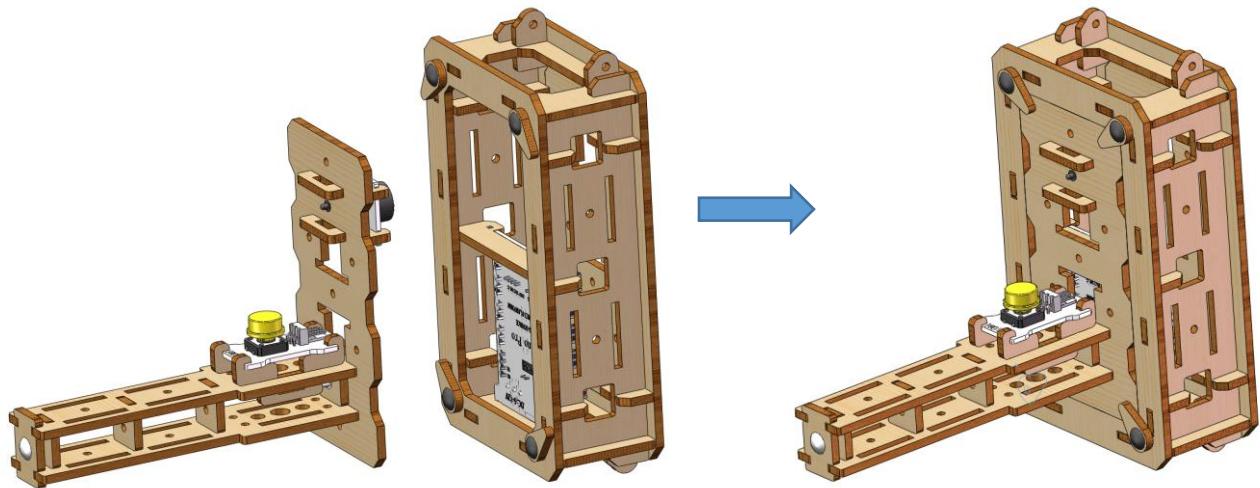
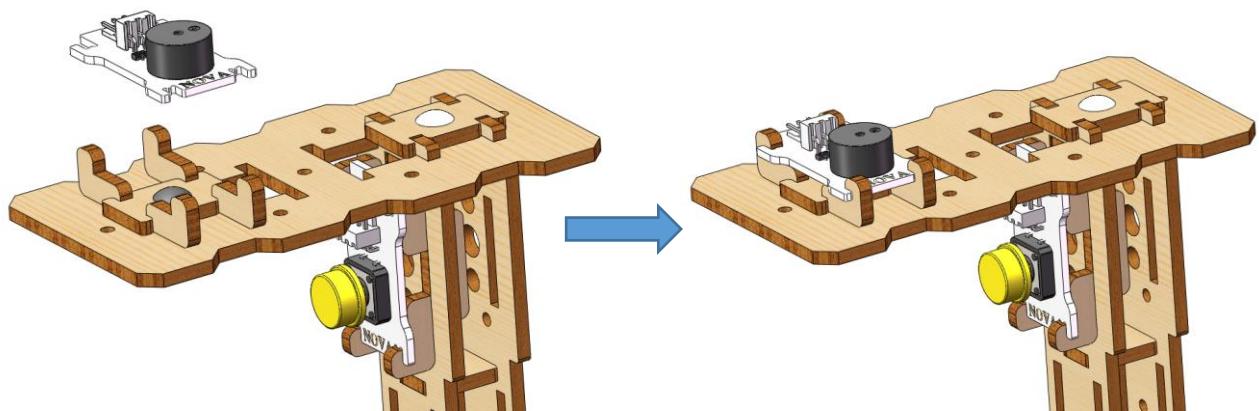
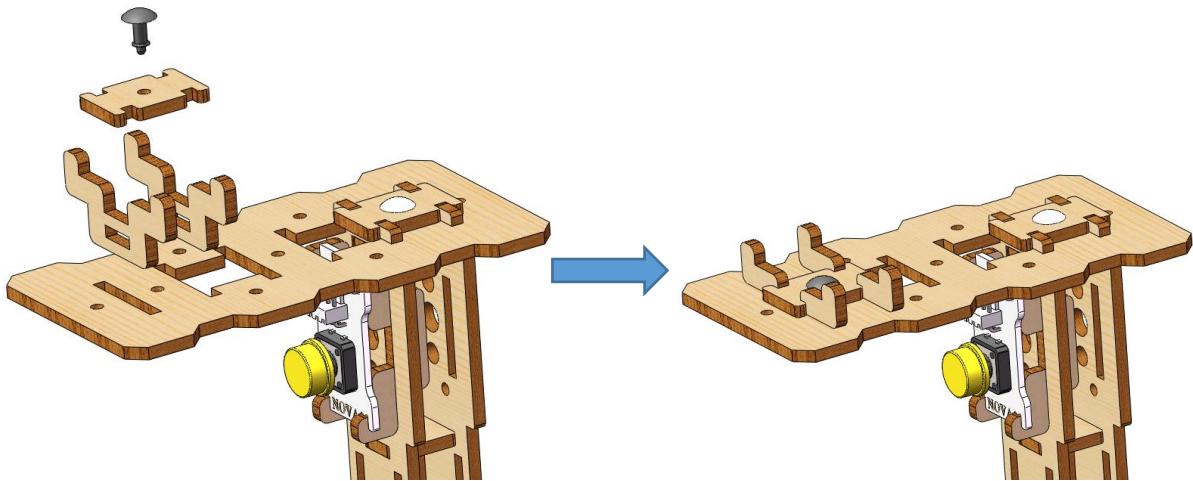
抽签仪、摇号机

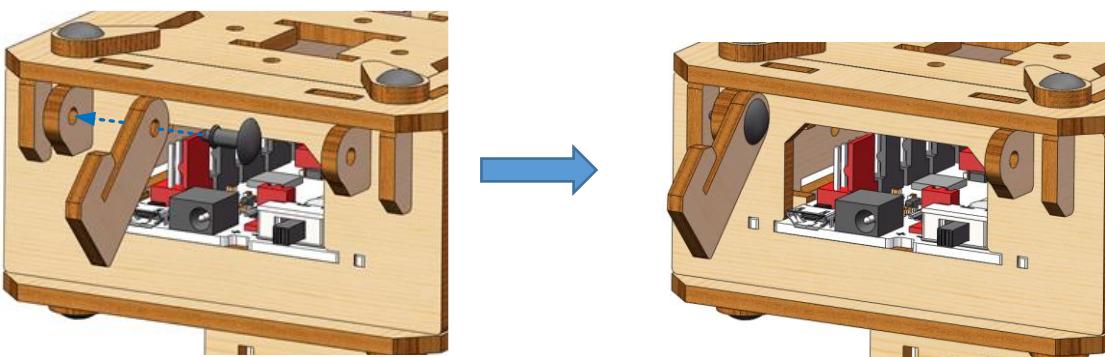
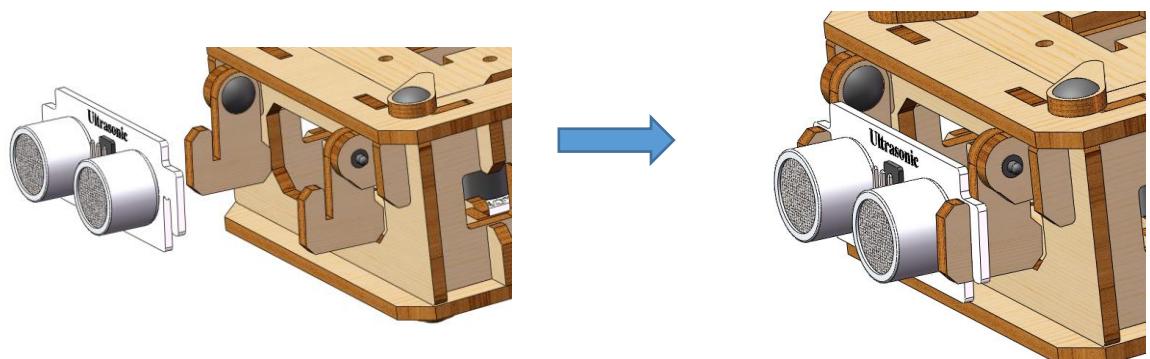
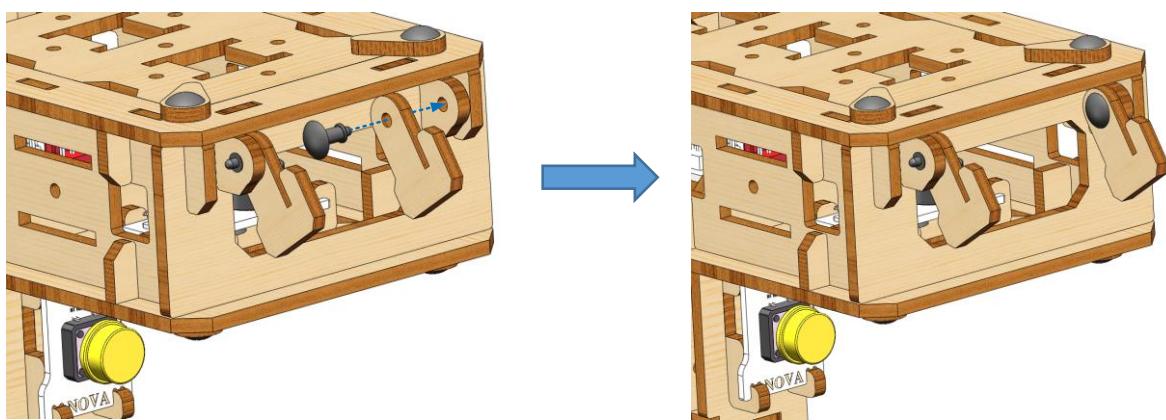
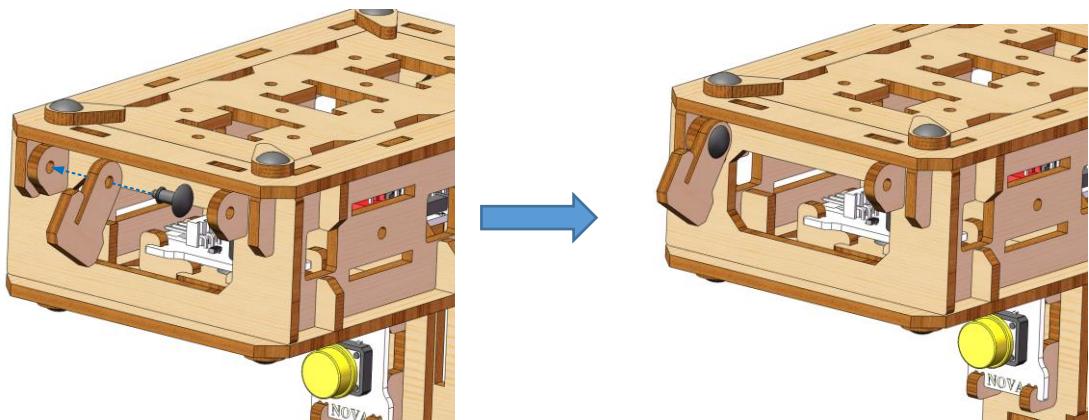


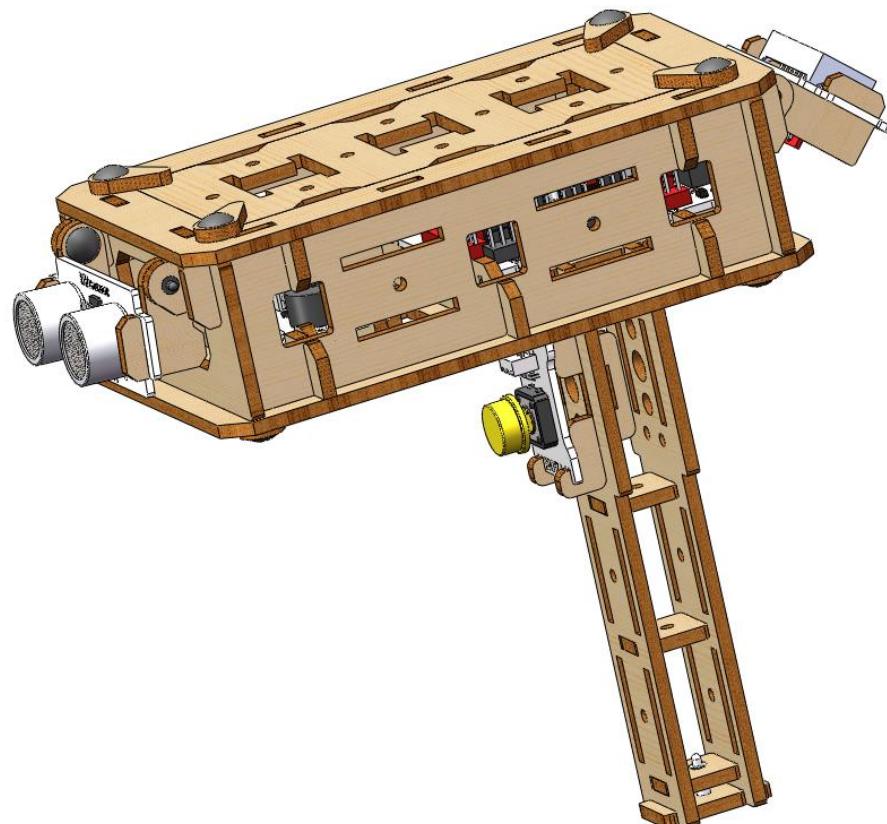
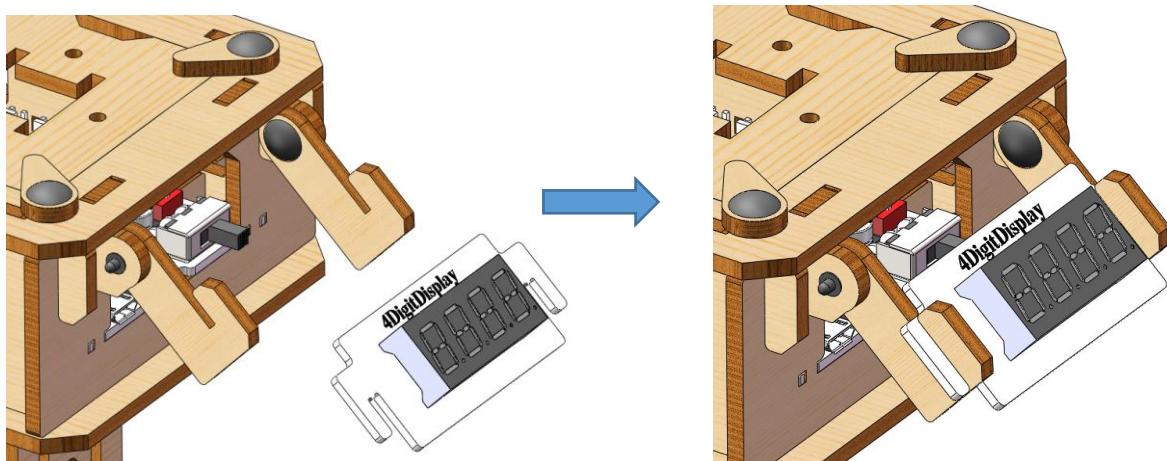
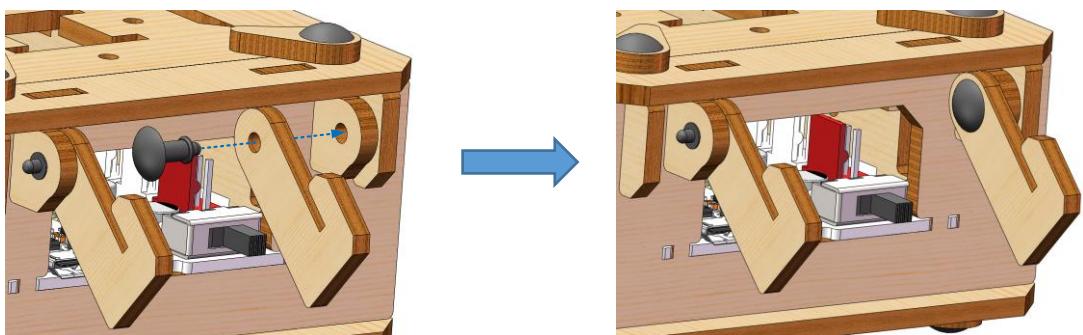


超声波测距仪

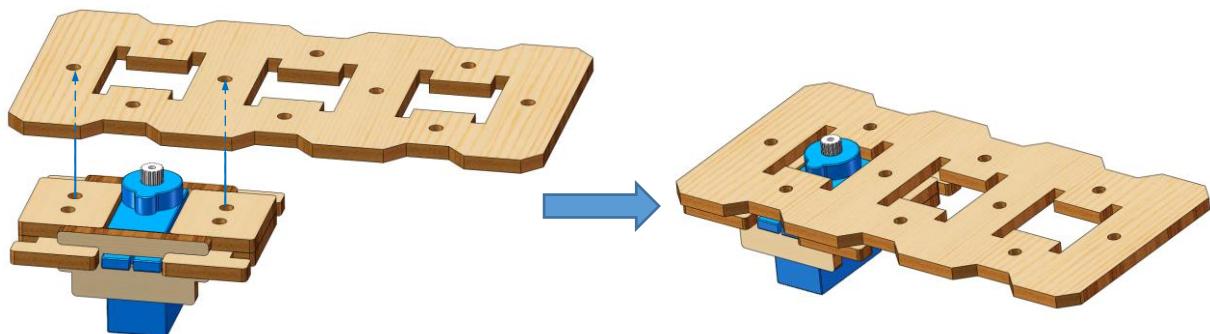
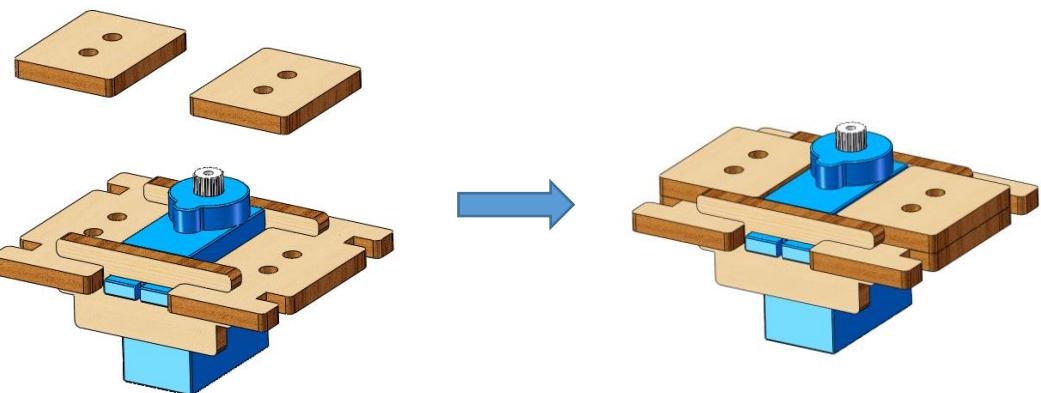
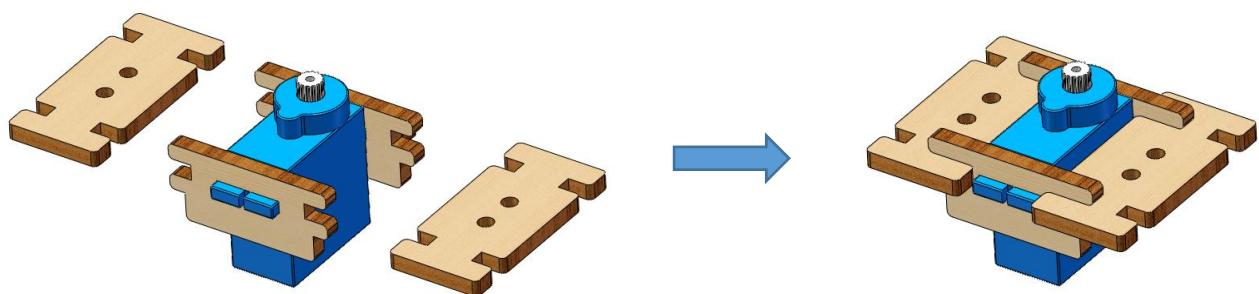
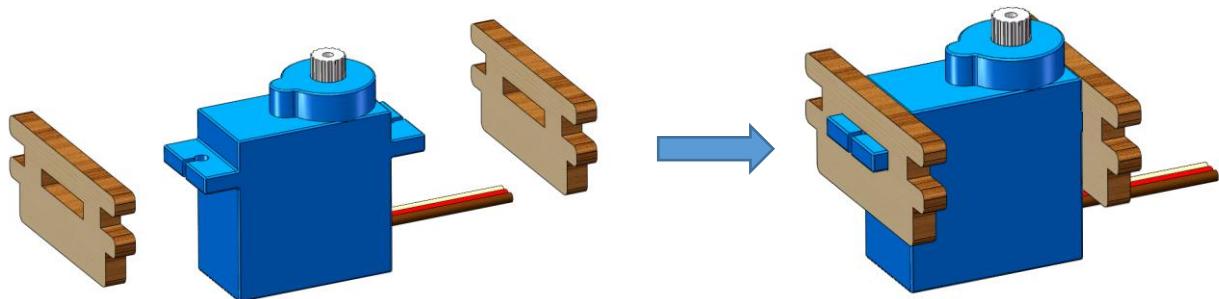


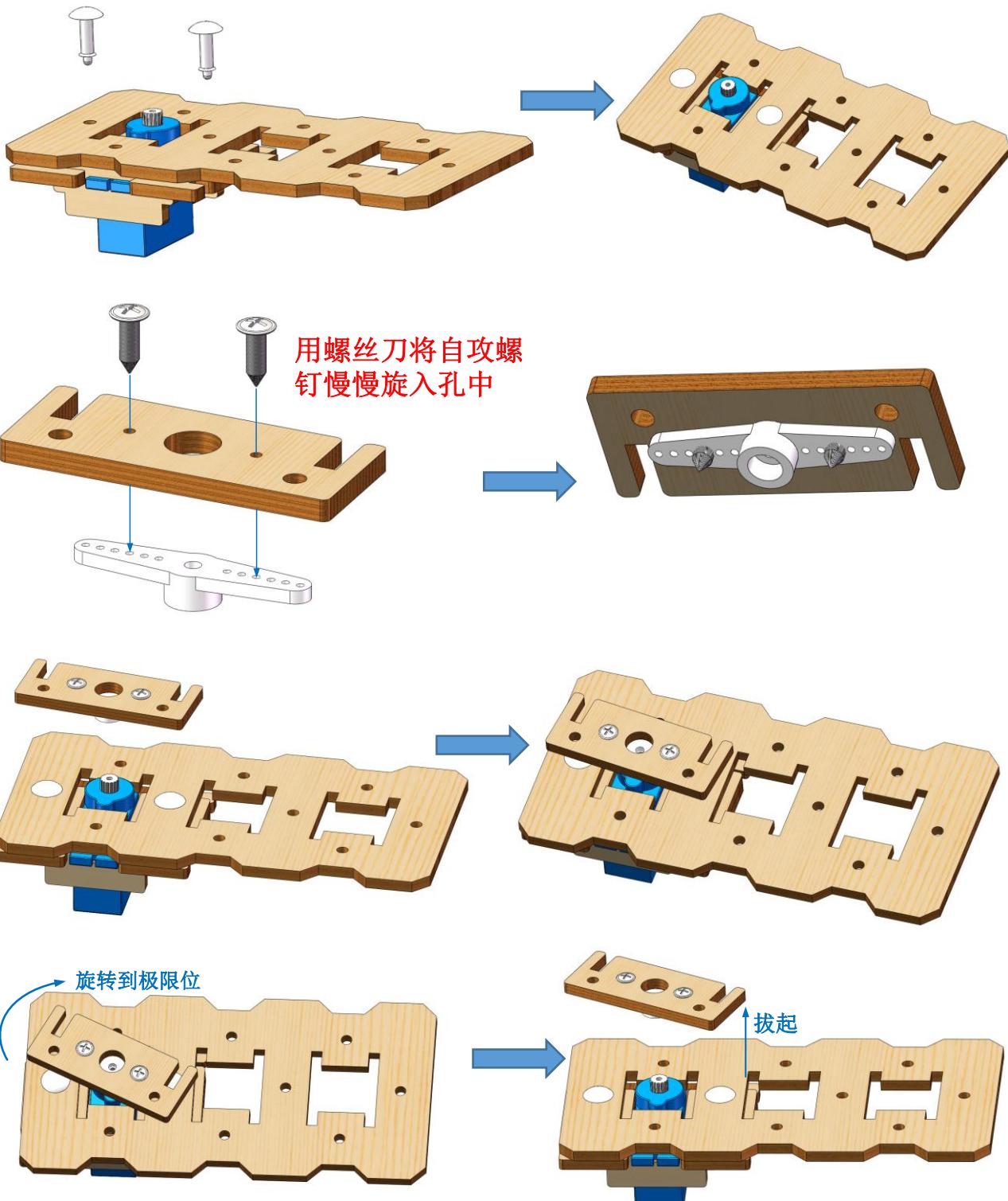


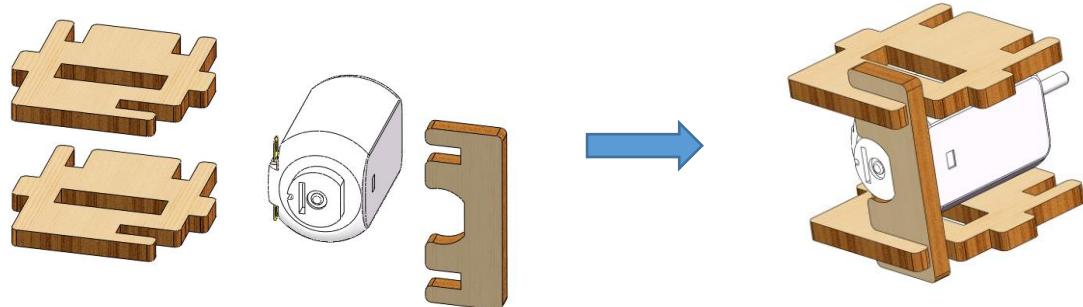
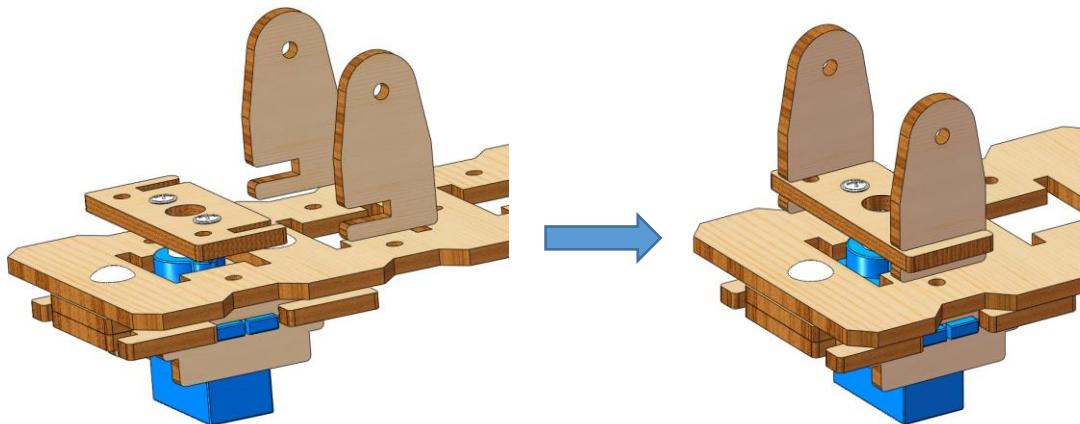
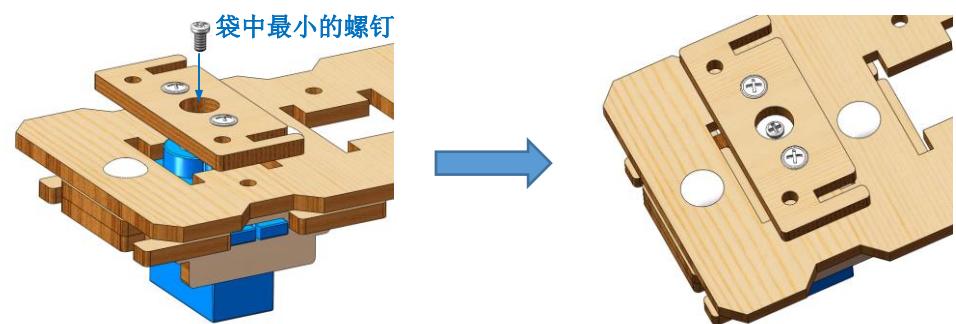
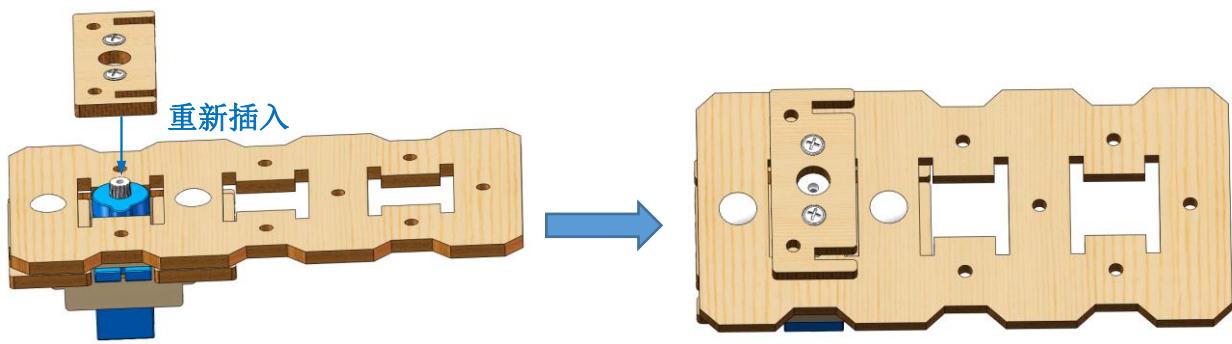


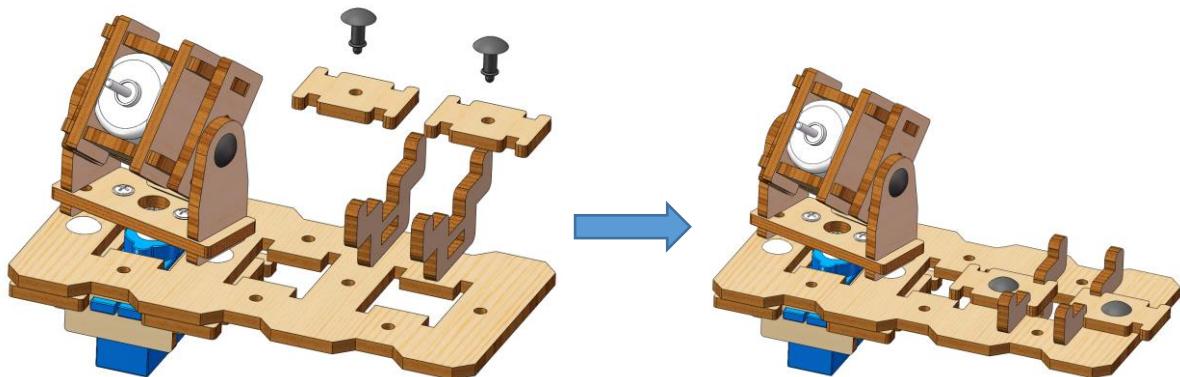
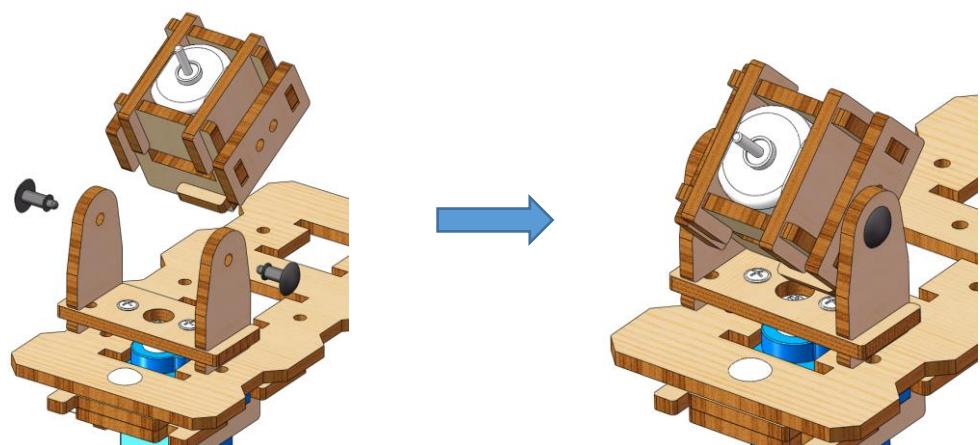
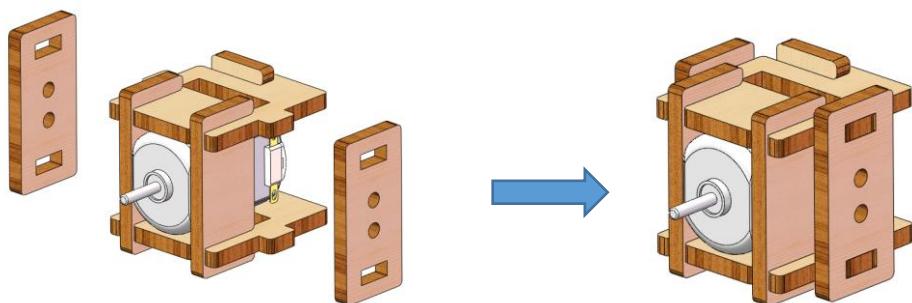
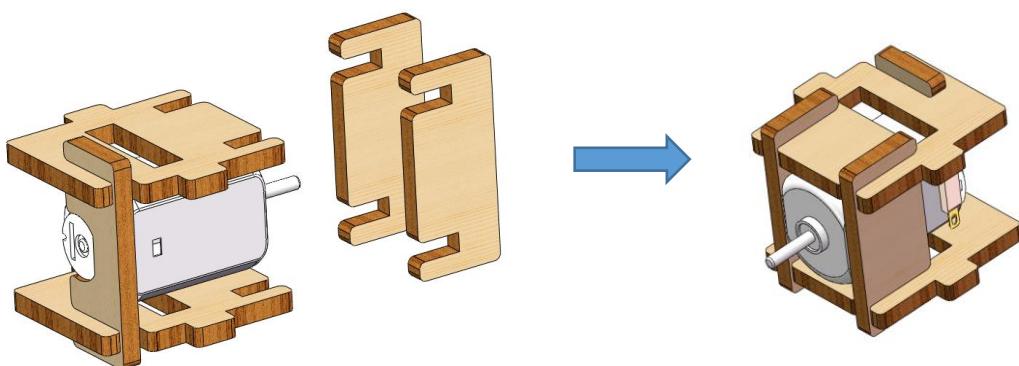


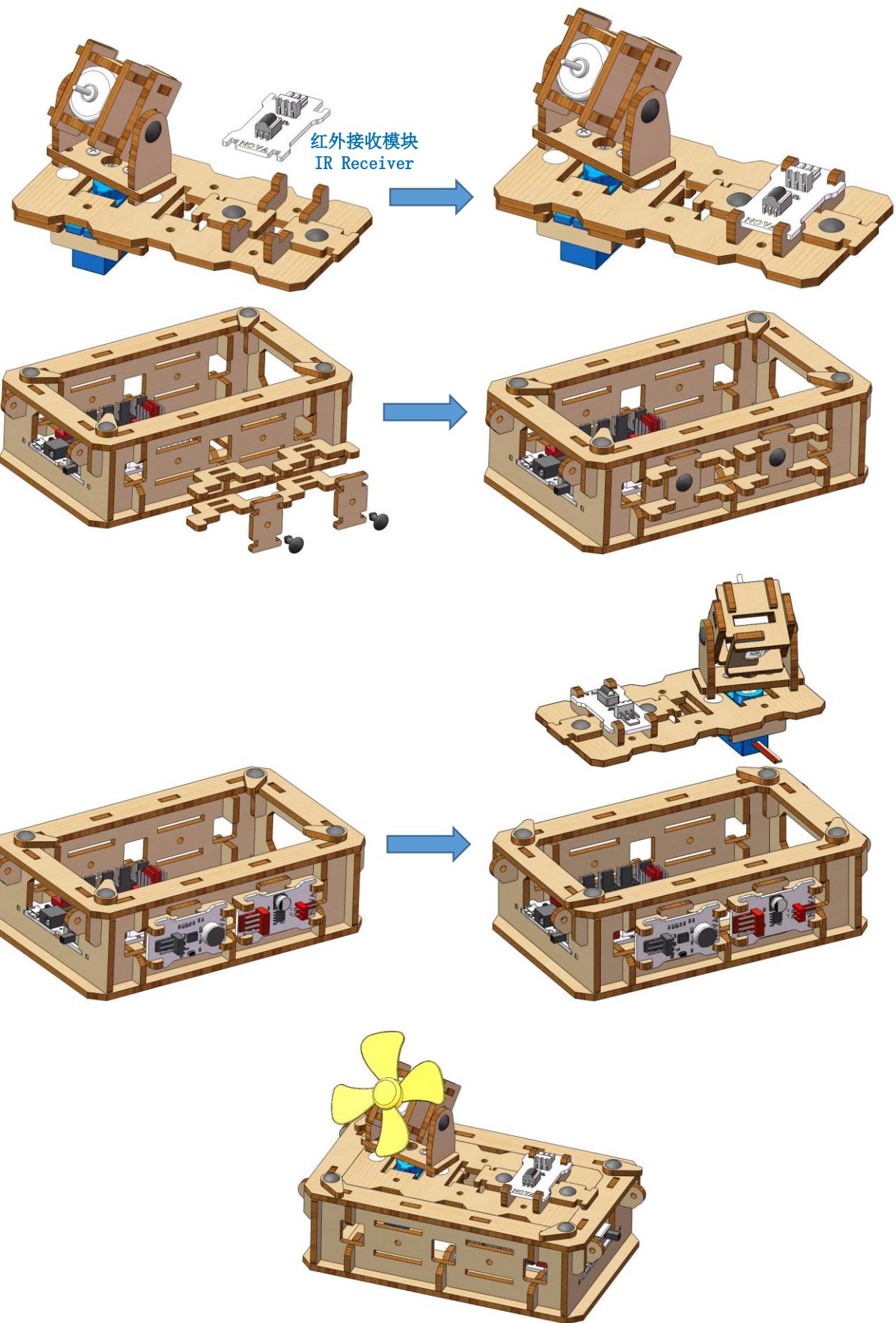
多功能风扇





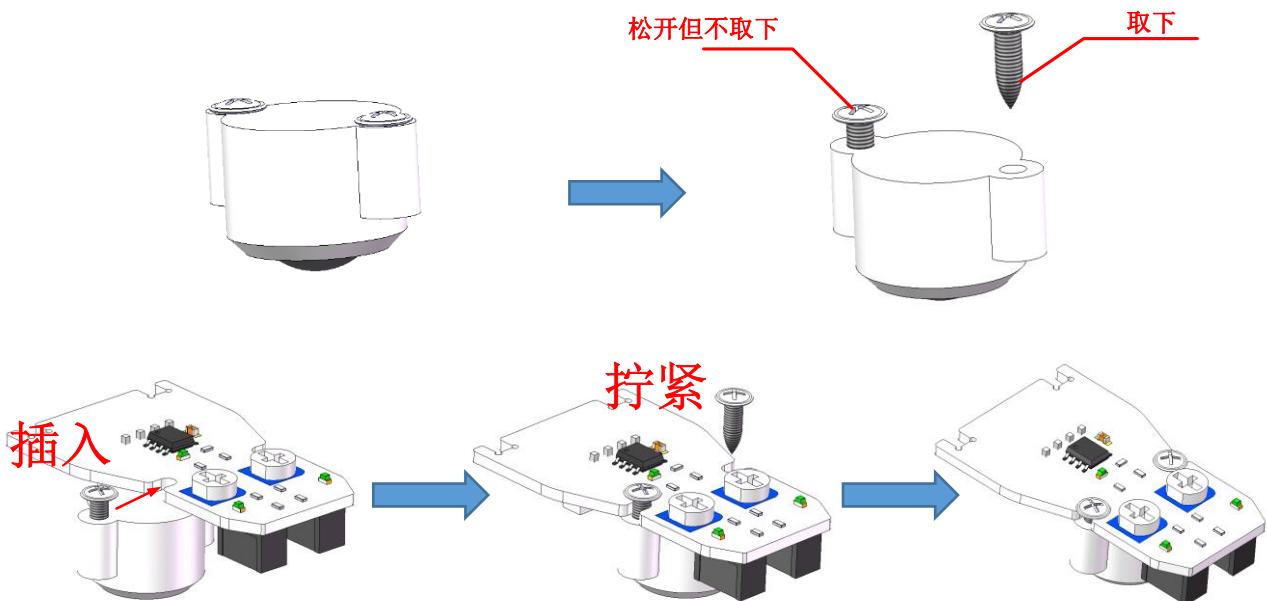




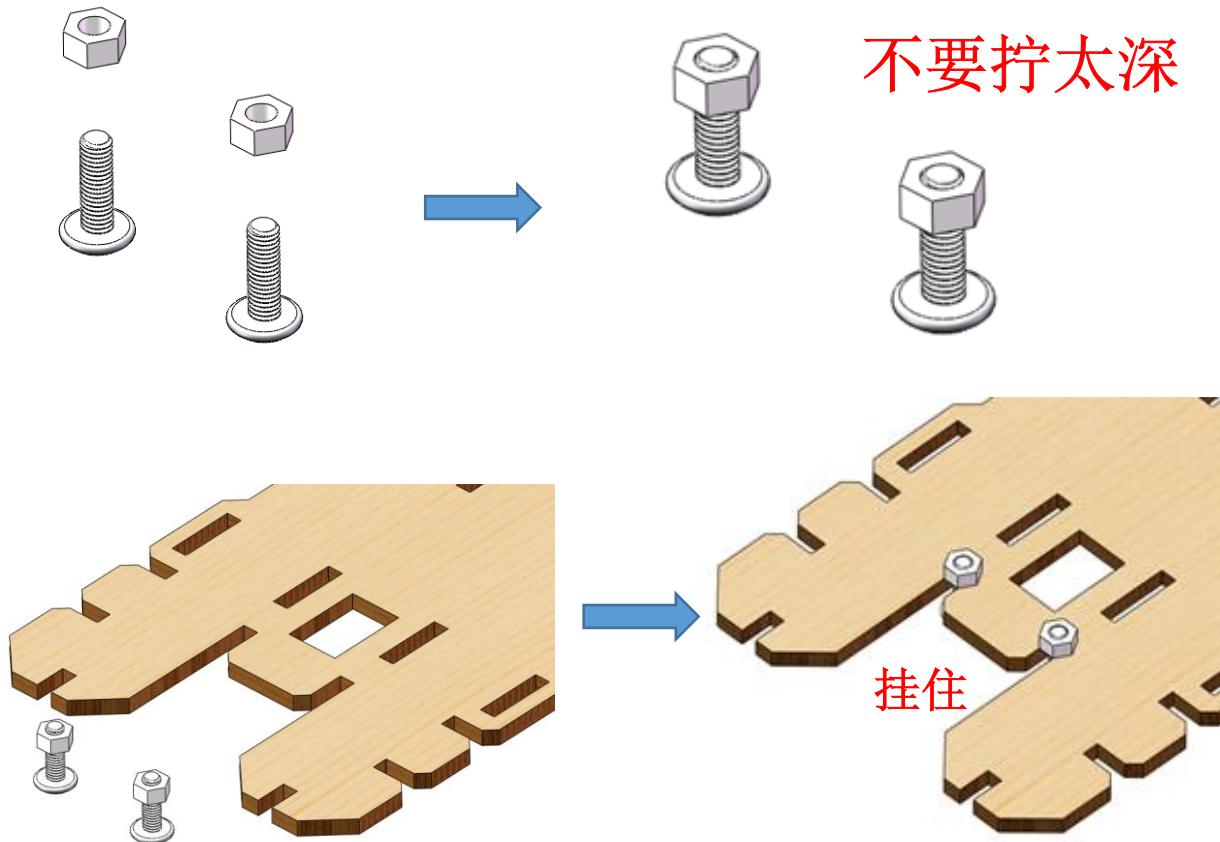


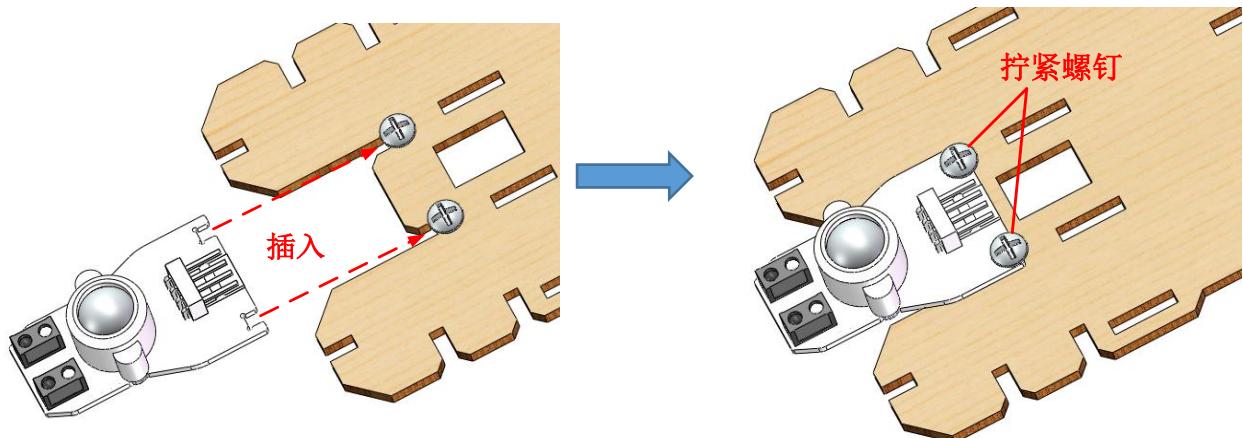
附录二 智能小车拼装说明书

1. 安装万向轮

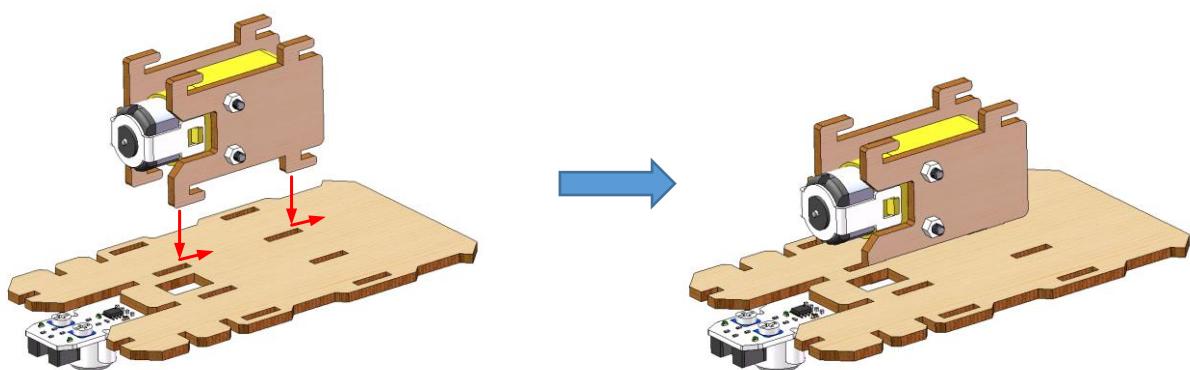
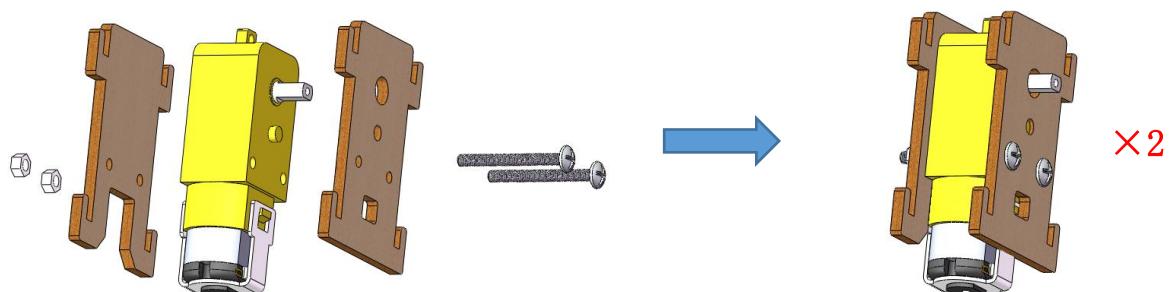


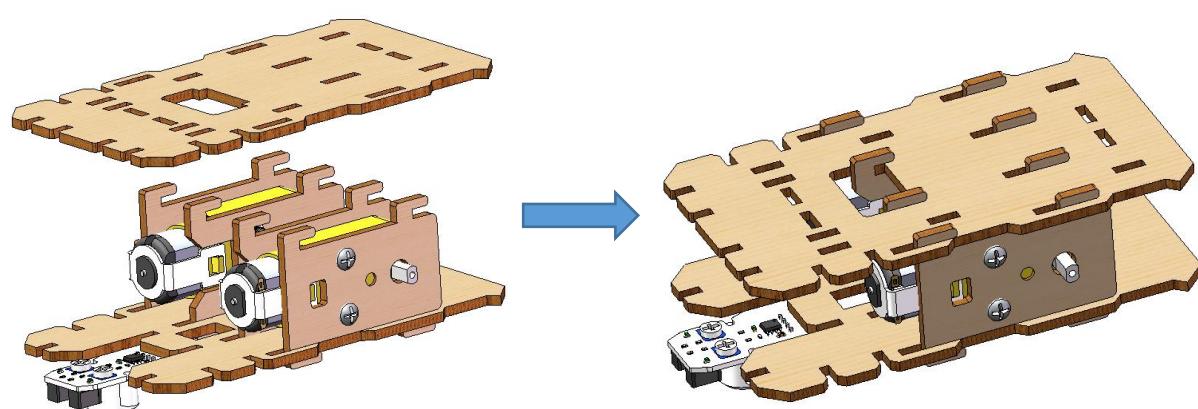
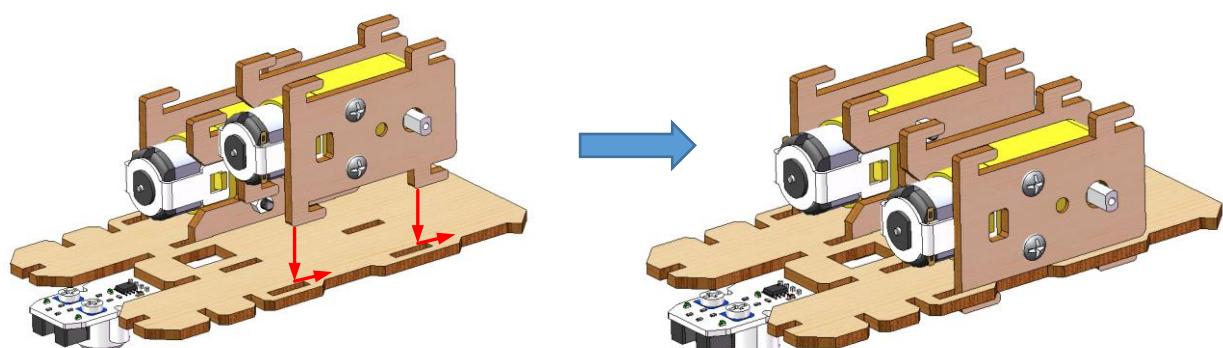
2. 安装巡线模块



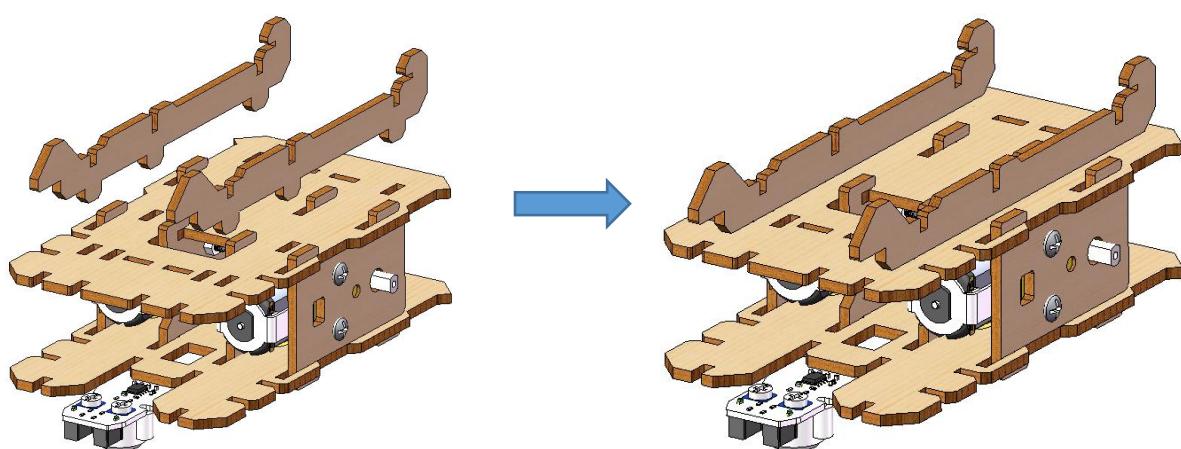


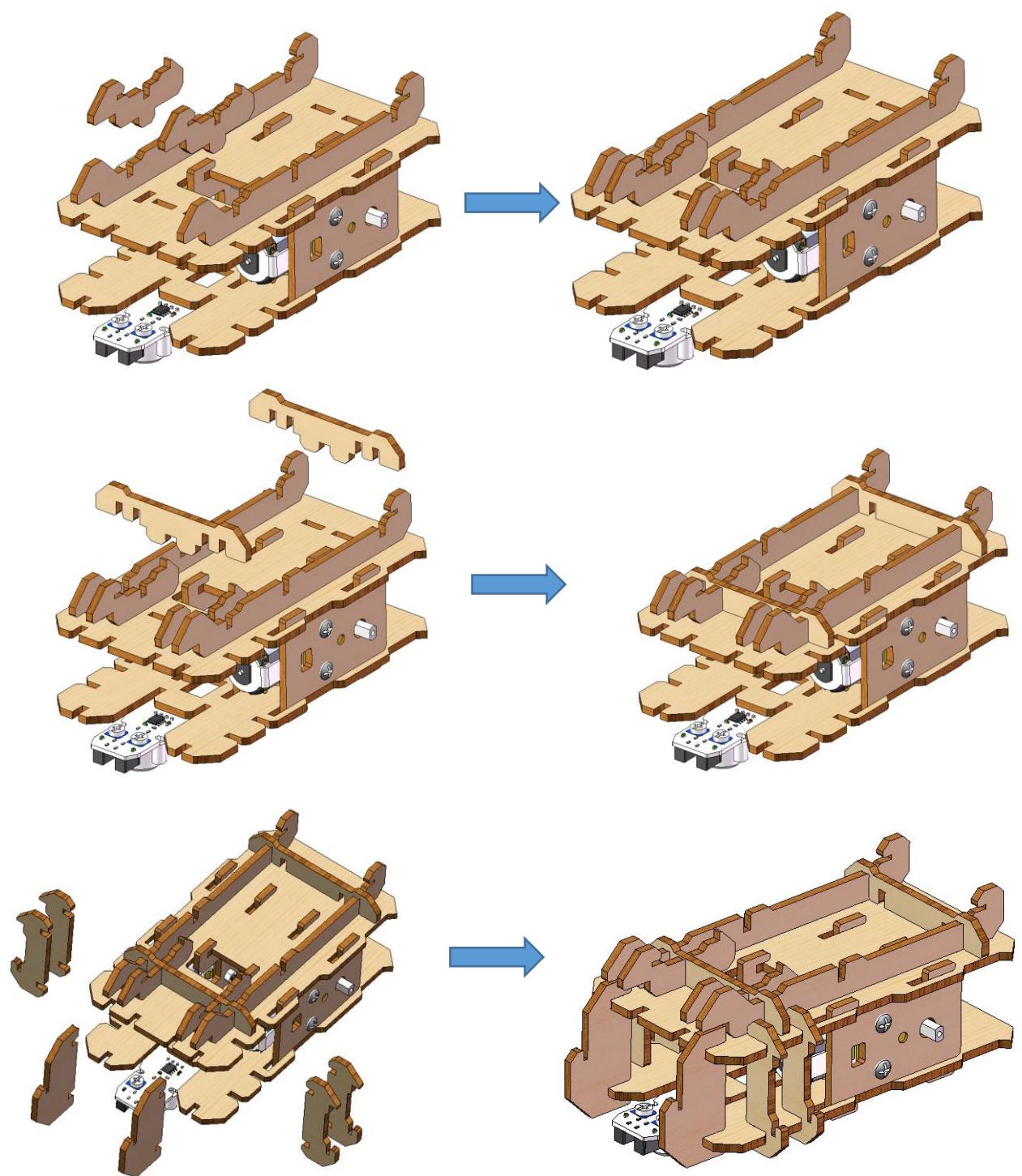
3. 安装电机



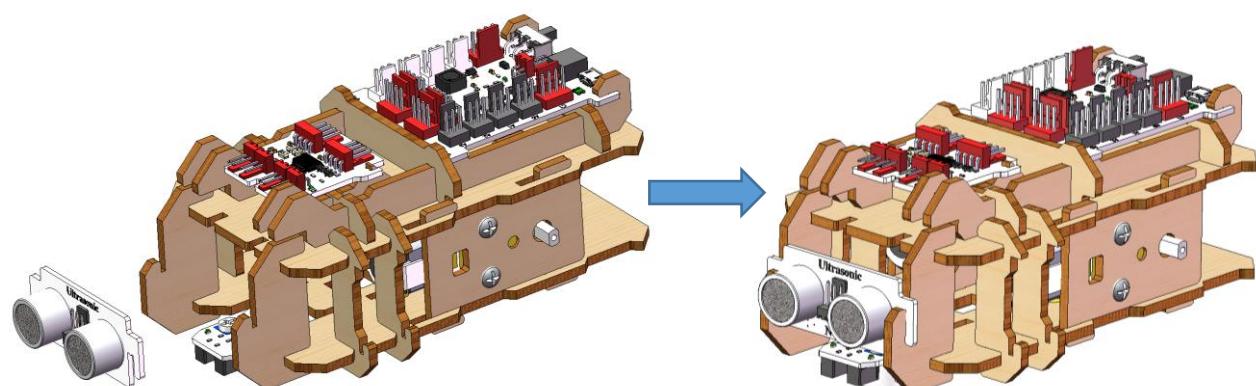
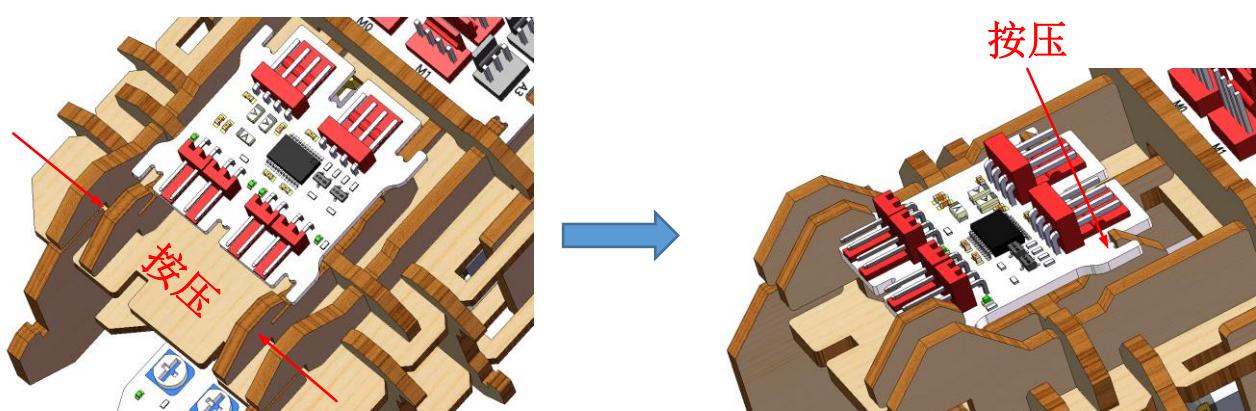
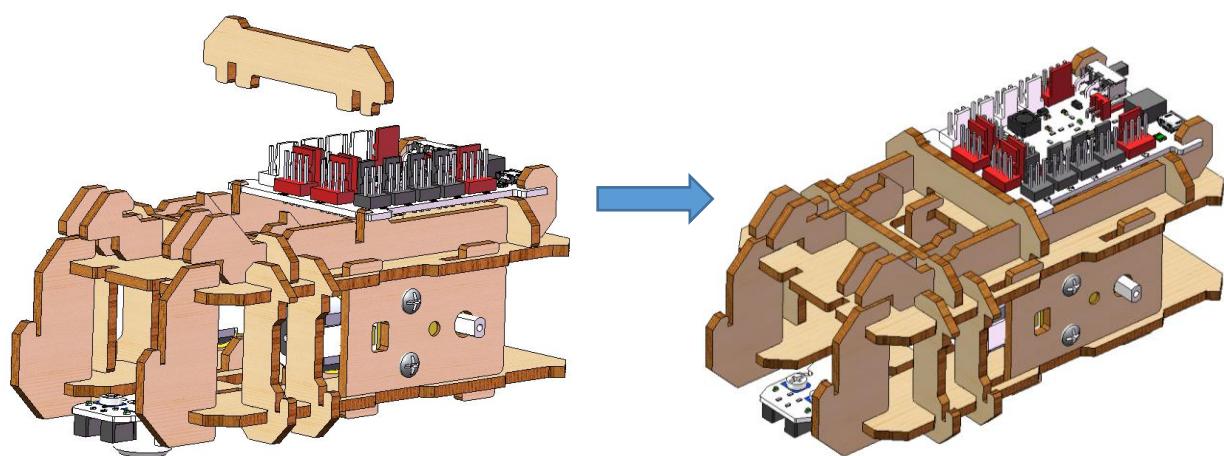
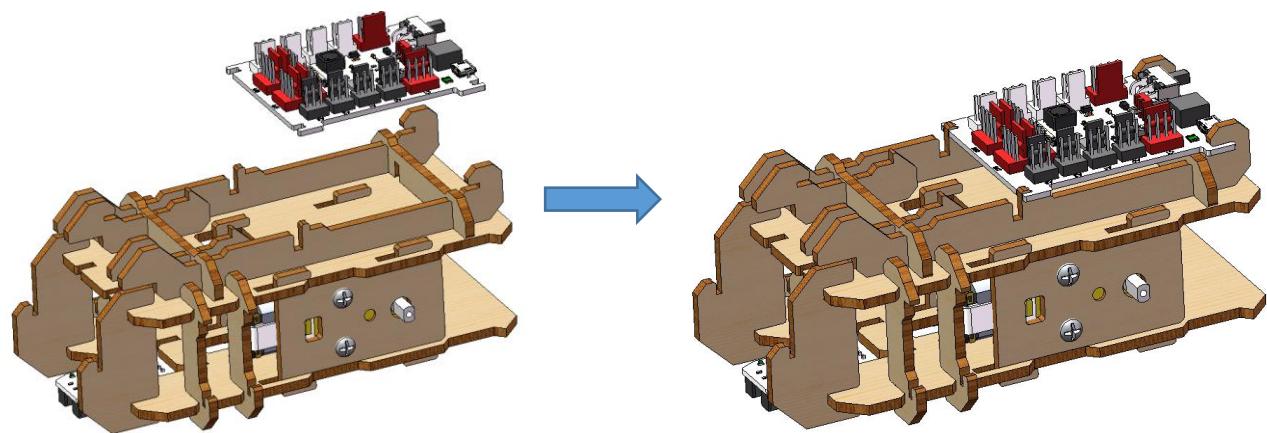


4. 安装外围框架





4. 安装电子模块



5. 安装轮胎

