00A0 2203∃ 2200∀ 2286⊆ 2713x 27FA⟺ 221A$\surd$ 221B$\sqrt[3]{}$ 2295⊕ 2297⊗ UTF8gbsn

SunFounder uno-and-mega-kit

www.sunfounder.com

2024 年 02 月 02 日

Contents

这是一个 Arduino 的基础学习套件，有 2 个版本分别适用*Arduino 项目 - Uno R3* 和*Arduino 项目 - Mega2560*。

里面包含了种类繁多的器件，比如电阻，电容，二极管/三极管，按键等简单器件，也包含一些传感器模块，比如光敏电阻，热敏电阻，超声波模块，温湿度传感器模块等。还有一些显示类的器件，比如 LED，7 段数码管，LCD1602 等。让你可以学习各种各样的电子电路知识。

课程顺序是从简单到复杂，建议你按顺序阅读每一章节。每一节课都包含了本节课的知识概括，所需器件图示及介绍链接，原理图及原理讲解，实验步骤，代码及代码分析等，让你可以清晰明白的掌握每一个器件使用和知识点。

另外还包含了图形化编程课程 - *Scratch 项目*，让你可以在制作一个动画的同时掌握元器件的使用，好玩又好学。

如果您想学习我们没有的其他项目，请随时发送电子邮件，我们将尽快更新我们的在线教程，欢迎任何建议。

这是电子邮件：service@sunfounder.com。

CHAPTER 1

套件中包含的器件

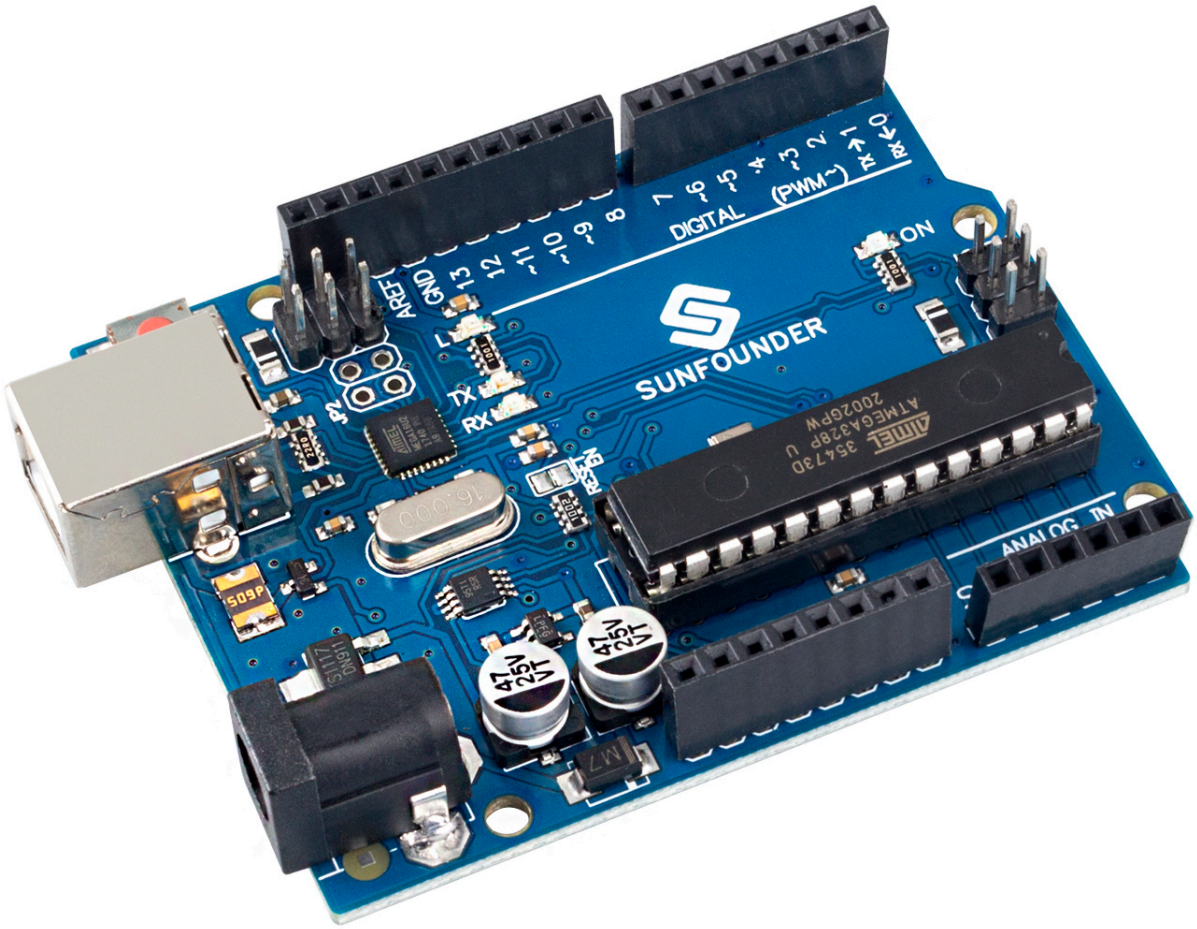
CHAPTER 2

器件介绍

下面是各个组件的介绍，里面包含了组件的运行原理和对应的项目。

通用器件

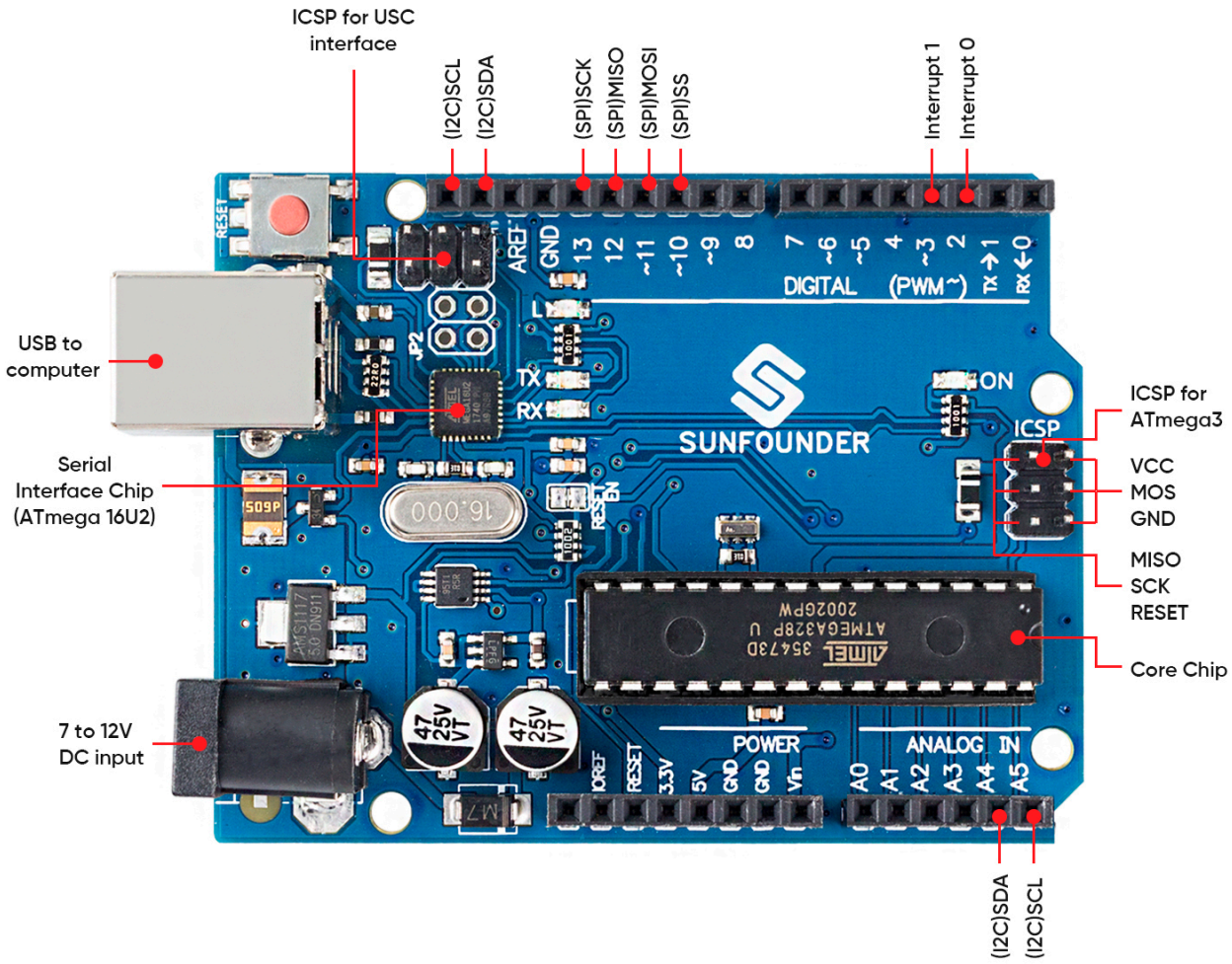
2.1 SunFounder R3 板



备注： SunFounder R3 板是一块与 Arduino Uno 功能几乎相同的主板，两块板可以互换使用。

SunFounder R3 板是基于 ATmega328P（数据表）的微控制器板。它有 14 个数字输入/输出引脚（其中 6 个可用作 PWM 输出）、6 个模拟输入、一个 16 MHz 陶瓷谐振器 (CSTCE16M0V53-R0)、一个 USB 连接、一个电源插孔、一个 ICSP 接头和一个复位按钮。它包含支持微控制器所需的一切；只需使用 USB 电缆将其连接到计算机或使用 AC-DC 适配器或电池为其供电即可开始使用。

技术参数



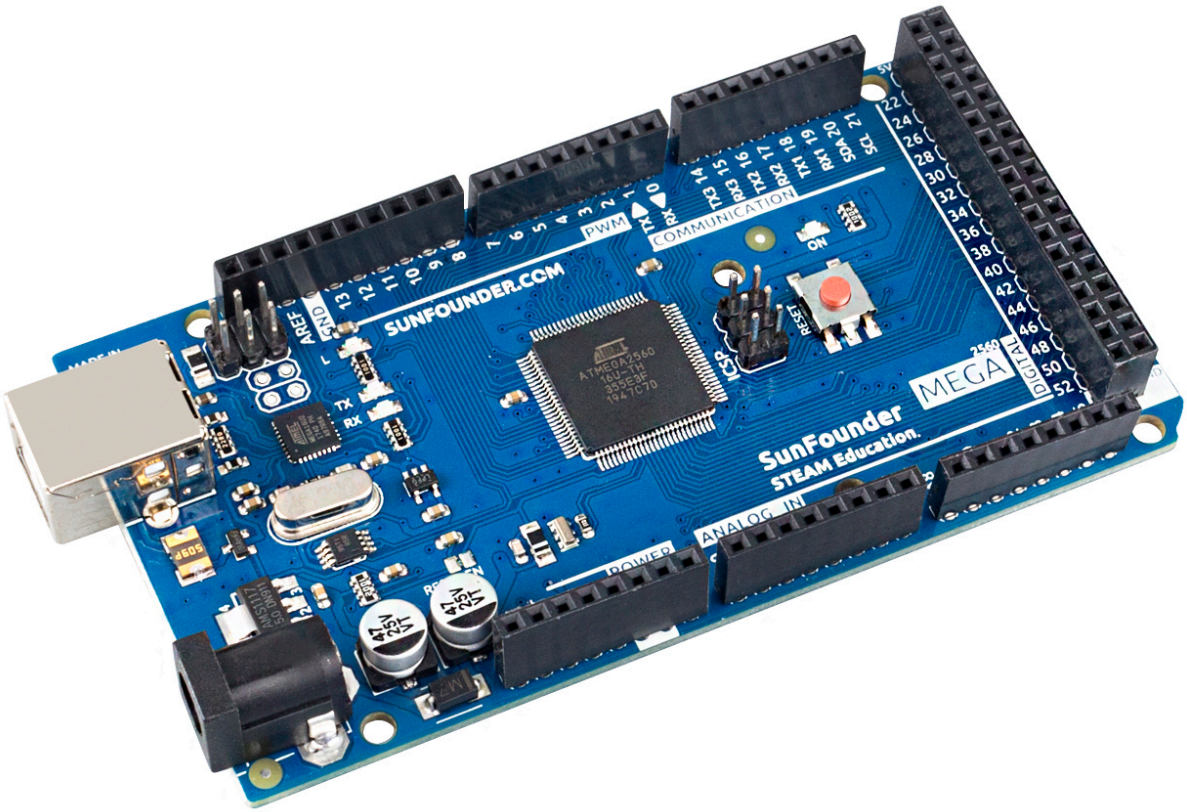
- 微控制器：ATmega328P
- 工作电压：5V
- 输入电压（推荐）：7-12V
- 输入电压（限制）：6-20V
- 数字 I/O 引脚: 14 (0-13, 其中 6 个提供 PWM 输出 (3, 5, 6, 9-11))
- PWM 数字 I/O 引脚：6（3、5、6、9-11）
- 模拟输入引脚：6 (A0-A5)
- 每个 I/O 引脚的直流电流：20 mA
- 3.3V 引脚的直流电流：50 mA
- 闪存：32 KB (ATmega328P) 其中 0.5 KB 由引导加载程序使用
- SRAM：2 KB (ATmega328P)
- EEPROM：1 KB (ATmega328P)
- 时钟速度：16 MHz
- 内置 LED: 13
- 长度：68.6 毫米

- 宽度：53.4 毫米
- 重量：25 克
- I2C 端口：A4(SDA)、A5(SCL)

更多

- Arduino IDE
- Arduino 编程语言参考
- *安装和介绍 Arduino IDE*
- ATmega328P 数据表

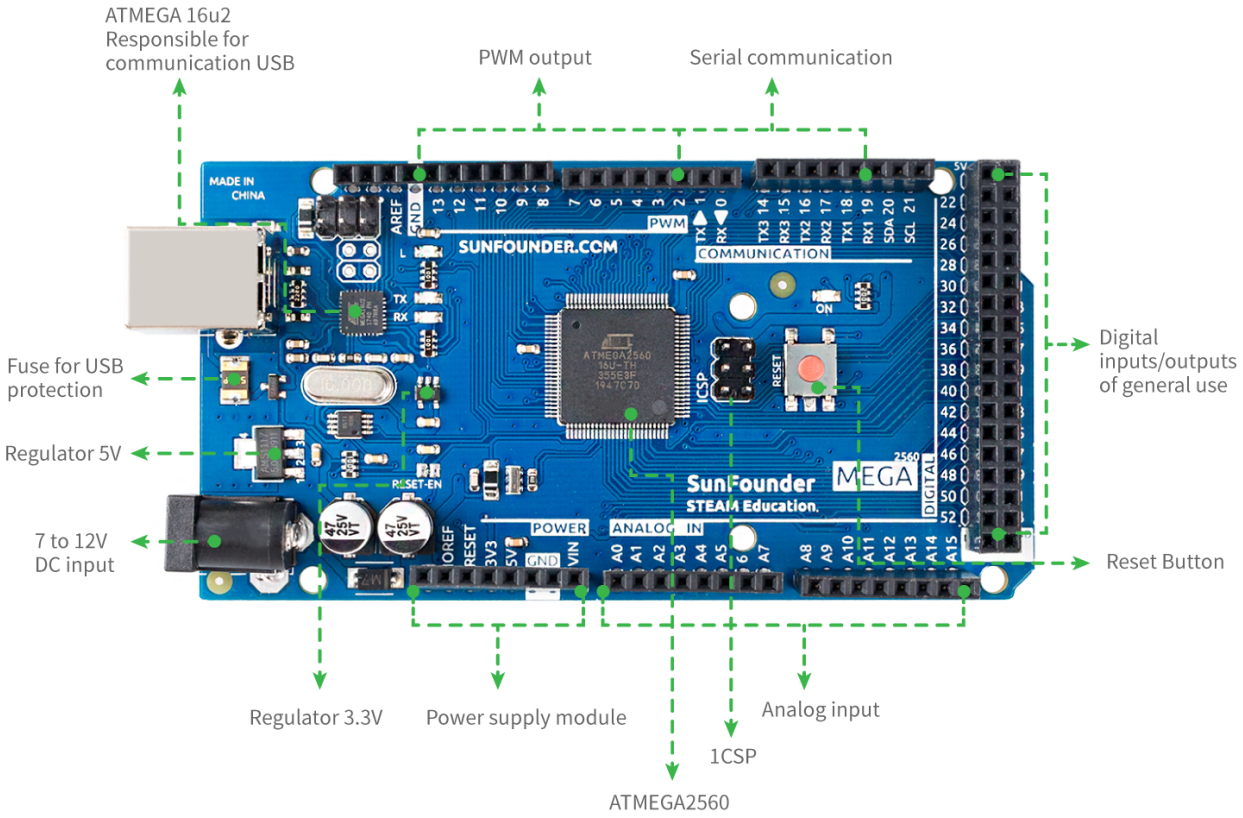
2.2 SunFounder Mega 板



备注： SunFounder Mega 板是一块与 Arduino Mega 2560 Rev3 功能几乎相同的主板，两块板可以互换使用。

SunFounder Mega Board 是基于 ATmega2560（数据表）的微控制器板。它具有 54 个数字输入/输出引脚（其中 15 个可用作 PWM 输出）、16 个模拟输入、4 个 UART（硬件串行端口）、一个 16 MHz 晶体振荡器、一个 USB 连接、一个电源插孔、一个 ICSP 接头、和一个重置按钮。它包含支持微控制器所需的一切；只需使用 USB 电缆将其连接到计算机或使用 AC-DC 适配器或电池为其供电即可开始使用。SunFounder Mega Board 板与大多数为 Uno 和之前的板 Duemilanove 或 Diecimila 设计的防护板兼容。

技术参数

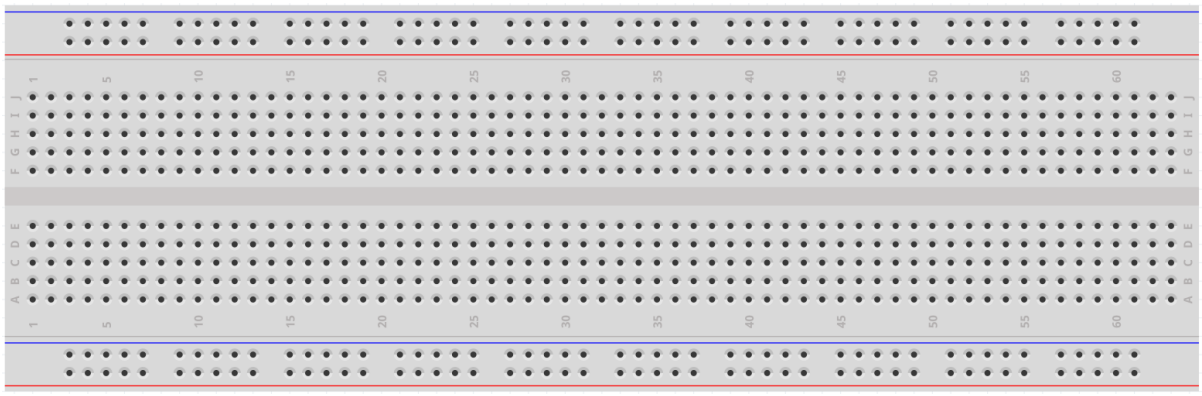


- 微控制器：ATmega2560
- 工作电压：5V
- 输入电压（推荐）：7-12V
- 输入电压（限制）：6-20V
- 数字 I/O 引脚 54 (0-53, 其中 15 提供 PWM 输出 (2-13, 44-46))
- 模拟输入引脚：16 (A0-A15)
- 每个 I/O 引脚的直流电流：20 mA
- 3.3V 引脚的直流电流：50 mA
- 闪存：256 KB，其中 8 KB 由引导加载程序使用
- SRAM：8 KB
- EEPROM：4 KB
- 时钟速度：16 MHz
- 内置 LED: 13
- 长度：101.52 毫米
- 宽度：53.3 毫米
- 重量：37 克
- I2C 端口：A4(SDA)、A5(SCL)；20(SDA), 21(SCL)

更多

- Arduino IDE
- Arduino 编程参考
- *安装和介绍 Arduino IDE*
- ATmega2560 数据表

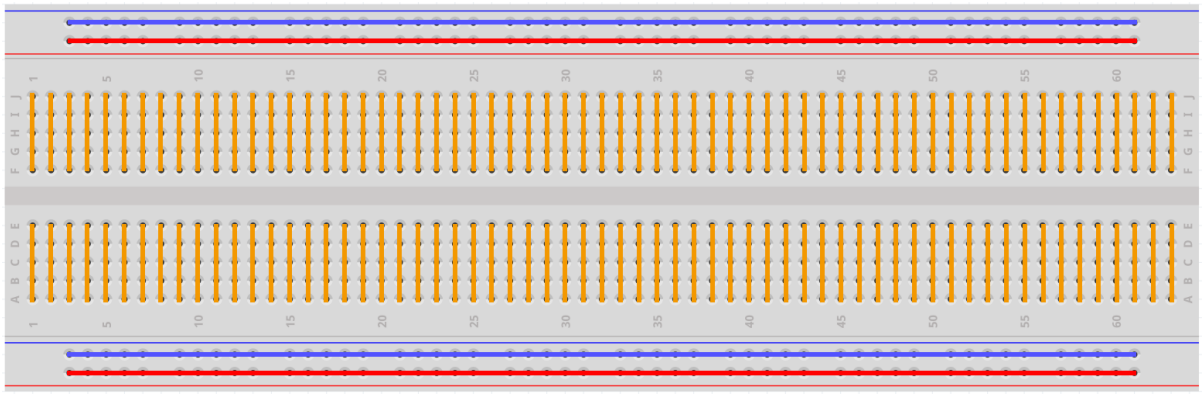
2.3 面包板



面包板是电子产品原型的构建基础。这个词最初指的是字面面包板，一种用于切片面包的抛光木头。[1] 在 1970 年代，无焊面包板（又名插件板，终端阵列板）问世，如今术语“面包板”通常用于指代这些。

它用于在完成任何电路设计之前快速构建和测试电路。它有许多孔，可以插入上述组件，如 IC 和电阻器以及跳线。面包板允许你轻松插入和移除组件。

图为面包板的内部结构。虽然面包板上的这些孔看起来是相互独立的，但它们内部实际上是通过金属条相互连接的。



如果你想了解更多关于面包板的信息，请参考：How to Use a Breadboard - Science Buddies

2.4 跳线

连接两个端子的导线称为跳线。有各种类型的跳线。在这里，我们专注于面包板中使用的那些。其中，它们用于将电信号从面包板上的任何位置传输到微控制器的输入/输出引脚。

跳线是通过将它们的“末端连接器”插入面包板中提供的插槽来安装的，在面包板的表面下方有几组平行板，根据区域以行或列的形式连接插槽。“末端连接器”插入面包板，无需焊接，位于特定原型中需要连接的特定插槽中。

跳线有母对母（F-F）、公对公 (M-M)、公对母 (M-F) 三种。我们之所以称其为“公对母”，是因为它的一端具有突出的尖端以及下沉的母端。公-公表示两端都是公头，而母-母表示两端都是母头。

备注:

- 在一个项目中可能会使用不止一种类型。
- 跳线的颜色不同，但并不代表它们的功能也相应不同；它只是为了更好地识别每个电路之间的连接而设计的。

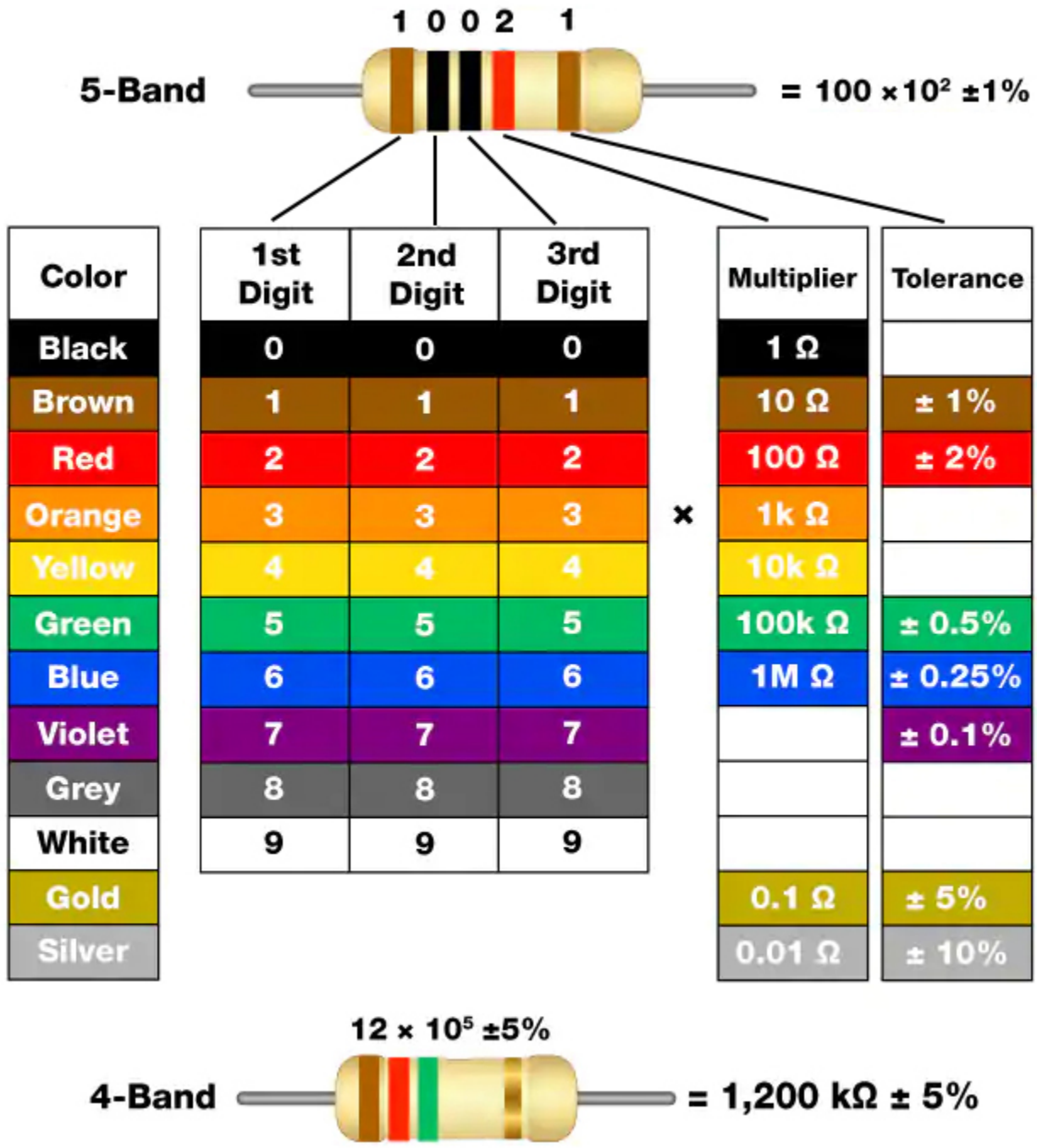
2.5 电阻

电阻器是一种可以限制支路电流的电子元件。固定电阻器是一种阻值不能改变的电阻器，而电位器或可变电阻器的电阻器是可以调节的。

电阻器的两种常用电路符号。通常，电阻标在其上。因此，如果你在电路中看到这些符号，则它代表电阻器。

Ω 是电阻的单位，较大的单位有 KΩ、MΩ 等，它们的关系可以表示为：1 MΩ=1000 KΩ，1 KΩ = 1000 Ω。通常，电阻值标在其上。

使用电阻时，首先要知道它的阻值。这里有两种方法：你可以观察电阻上的色环，或者用万用表测量电阻。建议你使用第一种方法，因为它更方便、更快捷。



如卡片所示，每种颜色代表一个数字。

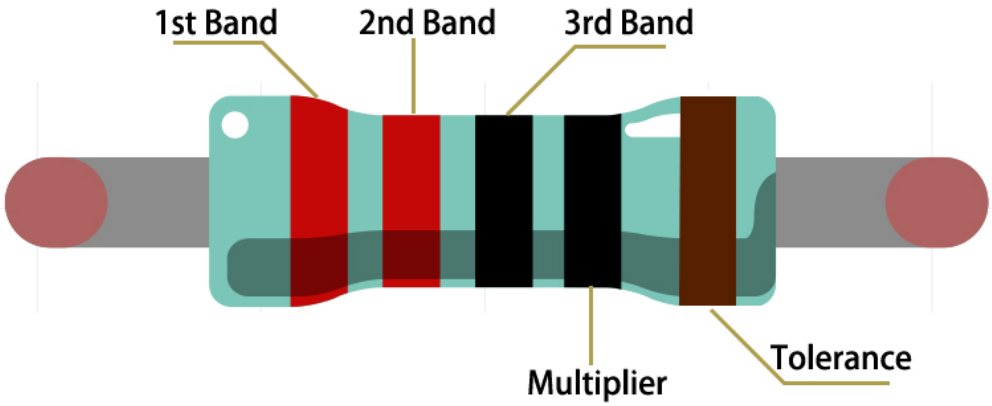
Black 黑	Brown 棕	Red 红	Orange 橙	Yellow 黄	Green 绿	Blue 蓝	Violet 紫	Grey 灰	White 白	Gold 金	Silver 银
0	1	2	3	4	5	6	7	8	9	0.1	0.01

通常说的 4 环，5 环和 6 环电阻指的是电阻上有多少种的颜色色环

通常，当你获得一个电阻器时，你可能会发现很难决定从哪一端开始读取颜色。提示是第 4 和第 5 个色环之间的差距会比较大。

因此，可以观察到电阻一端的两个色环之间的间隙；如果它比任何其他带隙都大，那么你可以从相反的一侧读取。

让我们看看如何读取 5 环电阻器的阻值，如下所示。



所以对于这个电阻，应该从左到右读取电阻。该值应采用以下格式：1st Band 2nd Band 3rd Band x 10^Multiplier (Ω) 并且允许误差为 ±Tolerance%。所以这个电阻的阻值为 2(red) 2(red) 0(black) x 10^0(black) Ω = 220 Ω，允许误差为 ±1%（棕色）。

常用电阻色环

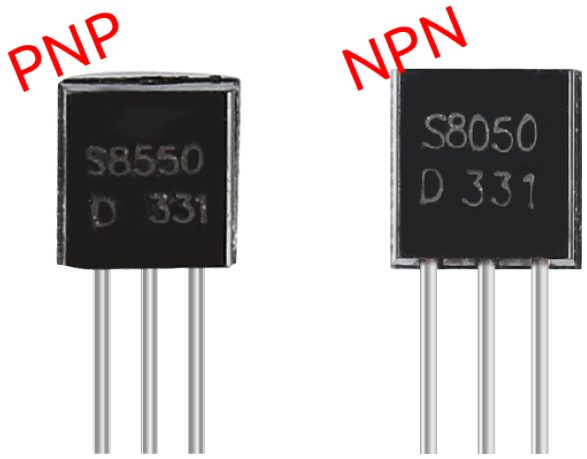
电阻	色环
10Ω	棕黑黑金棕
100Ω	棕黑黑黑棕
220Ω	红红黑黑棕
330Ω	橙橙黑黑棕
1kΩ	棕黑黑棕棕
2kΩ	红黑黑棕棕
5.1kΩ	绿棕黑棕棕
10kΩ	棕黑黑红棕
100kΩ	棕黑黑橙棕
1MΩ	棕黑黑绿棕

你可以从 Wiki：Resistor - Wikipedia 了解有关电阻器的更多信息。

示例

- *第 1 课闪烁的 LED* (Mega 板项目)
- *第 3 课按键* (Mega 板项目)
- *第 1 课闪烁的 LED* （R3 板项目）
- *第 3 课按键* （R3 板项目）
- *15. 游戏 - 吃苹果* （Scratch 项目）
- *2. 台灯* （Scratch 项目）

2.6 三极管



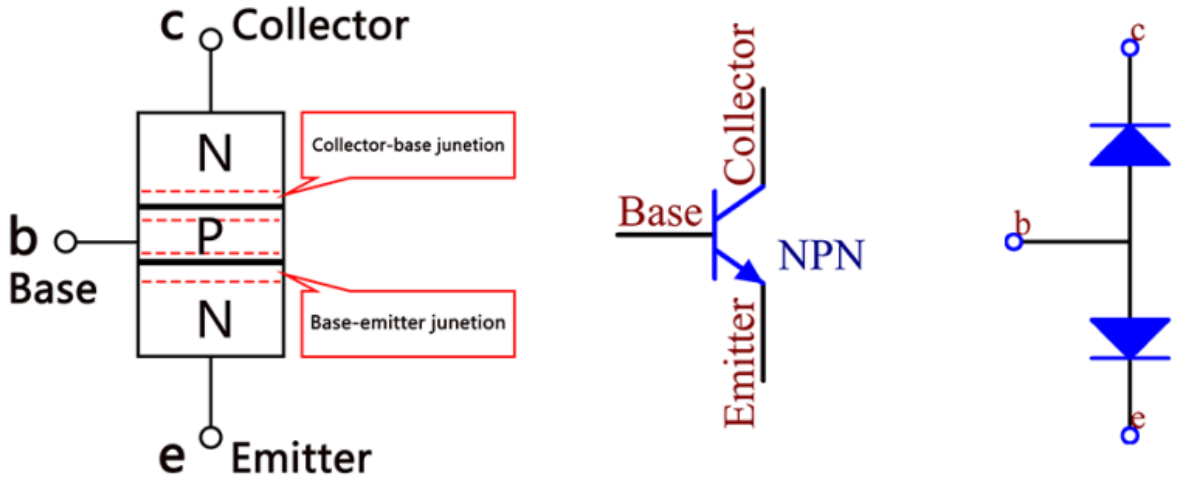
晶体管是一种通过电流控制电流的半导体器件。它的作用是将微弱信号放大为更大幅度的信号，也用于非接触式开关。

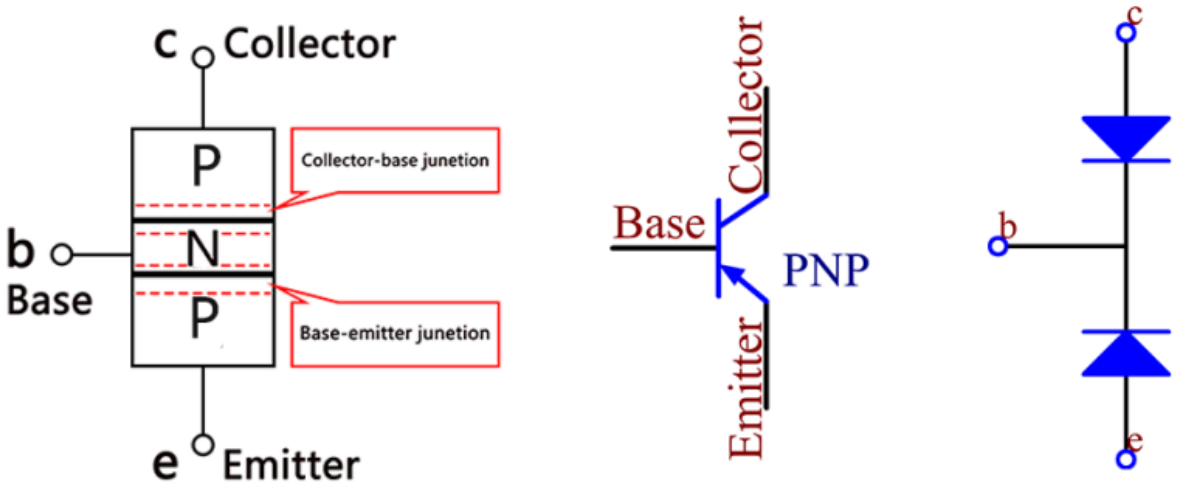
晶体管是由 P 型和 N 型半导体组成的三层结构。它们在内部形成三个区域。中间较薄的是基区；另外两个都是 N 型或 P 型的——具有强多数载流子的较小区域是发射极区，而另一个是集电极区。这种结构使晶体管成为放大器。从这三个区域分别产生三个极点，分别是基极（b）、发射极（e）和集电极（c）。它们形成两个 PN 结，即发射结和集电结。晶体管电路符号中箭头的方向表示发射结的方向。

- P-N 结 - 维基百科

根据半导体类型，晶体管可分为两类，NPN 和 PNP 型。从缩写可以看出，前者是由两个 N 型半导体和一个 P 型半导体组成，后者则相反。见下图。

备注： s8550 是 PNP 晶体管，s8050 是 NPN 晶体管，它们看起来非常相似，我们需要仔细检查才能看到它们的标签。





当高电平信号通过 NPN 晶体管时，它被激励。但是 PNP 需要一个低电平信号来管理它。两种类型的晶体管都经常用于非接触式开关，就像在这个实验中一样。

将标签面朝向我们，针脚朝下。从左到右的引脚是发射极（e）、基极（b）和集电极（c）。

- S8050 三极管数据表
- S8550 三极管数据表

2.7 电容

电容，是指在给定电位差下所储存的电荷量，记为 C，国际单位为法拉（F）。一般来说，电荷在电场中受力移动。当导体之间存在介质时，电荷的运动受到阻碍，电荷在导体上积聚，导致电荷积聚。

存储的电荷量称为电容。由于电容器是电子设备中应用最广泛的电子元件之一，因此广泛应用于直流隔离、耦合、旁路、滤波、调谐回路、能量转换、控制电路等。电容器分为电解电容器、固态电容器等。

根据材料特性，电容器可分为：铝电解电容器、薄膜电容器、钽电容器、陶瓷电容器、超级电容器等。

- 陶瓷电容器 - 维基百科
- 电解电容器 - 维基百科

陶瓷电容上有 103 或 104 的标签，代表电容值，103=10x10^3pF，104=10x10^4pF

单位换算

1F=10^3mF=10^6uF=10^9nF=10^12pF

示例

- *第 4 课蜂鸣器* (Mega 板项目)
- *第 4 课蜂鸣器* （R3 板项目）
- *7. 门铃* (Scratch 项目)
- *15. 游戏 - 吃苹果* (Scratch 项目)

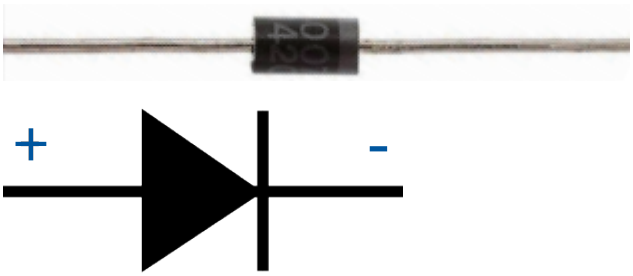
2.8 二极管

二极管是具有两个电极的电子元件。它只允许电流沿一个方向流动，这通常称为“整流”功能。因此，二极管可以被认为是单向阀的电子版本。

由于其单向导电性，二极管几乎用于所有具有一定复杂性的电子电路。它是最早的半导体器件之一，应用范围很广。

按用途分类可分为检波二极管、整流二极管、限幅二极管、稳压二极管等。

整流二极管



整流二极管是一种半导体二极管，用于使用整流桥应用将交流（交流）整流为直流（直流）。通过肖特基势垒的整流二极管的替代方案主要在数字电子产品中受到重视。该二极管能够传导从 mA 到几 kA 和高达几 kV 的电压值的电流值。

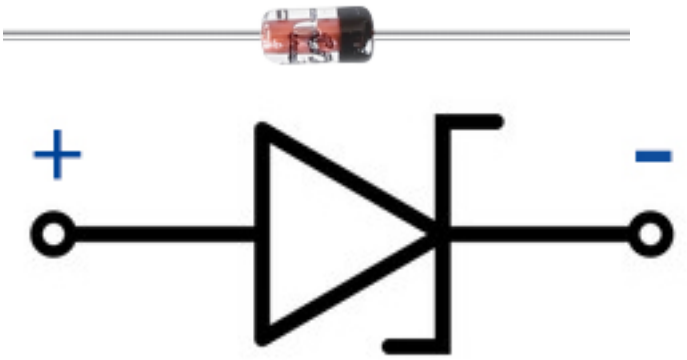
整流二极管的设计可以用硅材料完成，它们能够传导高电流值。这些二极管虽然名气不大，但仍使用锗或砷化镓基半导体二极管。Ge 二极管具有较小的允许反向电压以及较小的允许结温。与硅二极管相比，锗二极管的优点是在正向偏置下工作时阈值电压值低。

- 1N400x 通用二极管 - 维基百科

齐纳二极管

齐纳二极管是一种特殊类型的二极管，旨在在达到某个设定的反向电压（称为齐纳电压）时可靠地允许电流“反向”流动。

该二极管是一种半导体器件，在达到临界反向击穿电压时具有非常高的电阻。在这个临界击穿点，反向电阻降低到一个非常小的值，电流增加，而电压在这个低电阻区域保持恒定。



- 齐纳二极管 - 维基百科

示例

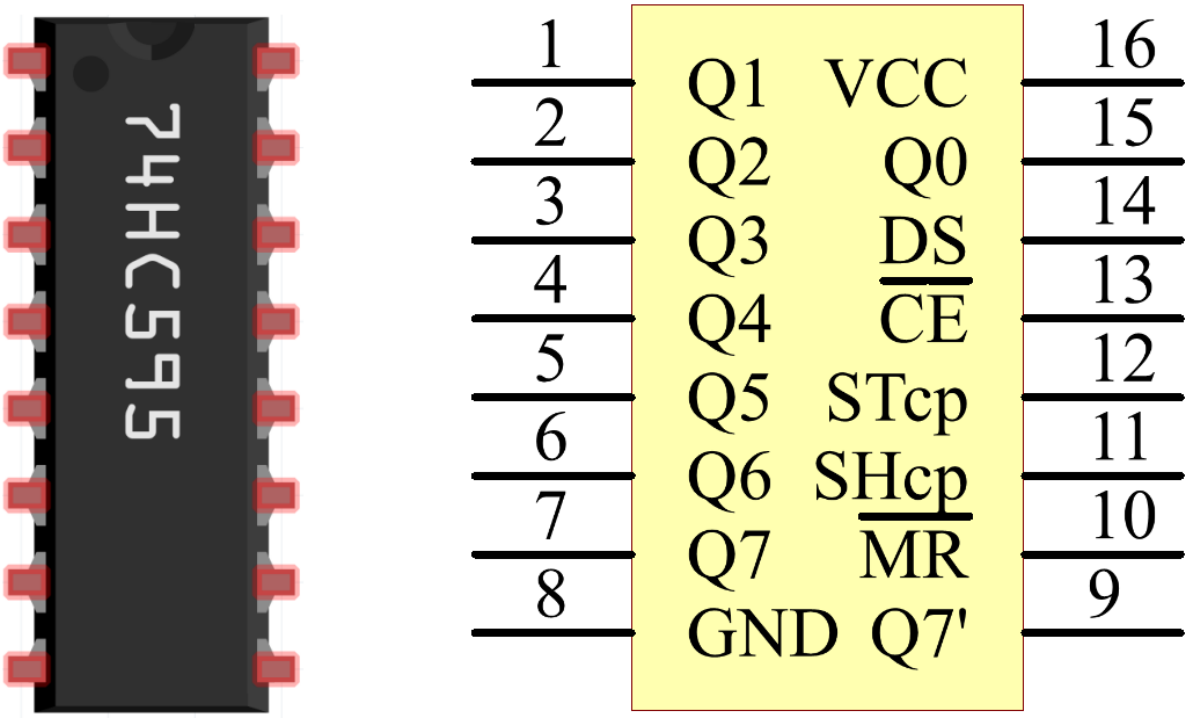
- 第 6 课继电器 (Mega 板项目)
- 第 6 课继电器 （R3 板项目）

芯片

2.9 74HC595

74HC595 由一个 8 位移位寄存器和一个具有三态并行输出的存储寄存器组成。它将串行输入转换为并行输出，因此你可以节省 MCU 的 IO 端口。当 MR(pin10) 为高电平，OE(pin13) 为低电平时，数据在 SHcp 的上升沿输入，通过 SHcp 的上升沿进入内存寄存器。如果两个时钟连接在一起，移位寄存器总是比内存寄存器早一个脉冲。内存寄存器中有一个串行移位输入引脚（Ds）、一个串行输出引脚（Q）和一个异步复位按钮（低电平）。内存寄存器输出一条 8 位并行、三种状态的总线。当 OE 使能（低电平）时，内存寄存器中的数据输出到总线。

- 74HC595 数据表



74HC595 的引脚及其功能：

- Q0-Q7：8 位并行数据输出引脚，可直接控制 8 个 LED 或 7 段显示器的 8 个引脚。
- Q7'：串联输出脚，接另一个 74HC595 的 DS，串联多个 74HC595
- MR：复位引脚，低电平有效；
- SHcp：移位寄存器的时序输入。在上升沿，移位寄存器中的数据连续移动一位，即 Q1 中的数据移动到 Q2，以此类推。在下降沿，移位寄存器中的数据保持不变。
- STcp：存储寄存器的时序输入。在上升沿，移位寄存器中的数据移动到内存寄存器中。
- CE：输出使能引脚，低电平有效。

- DS : 串行数据输入引脚
- VCC：正电源电压。
- GND：接地。

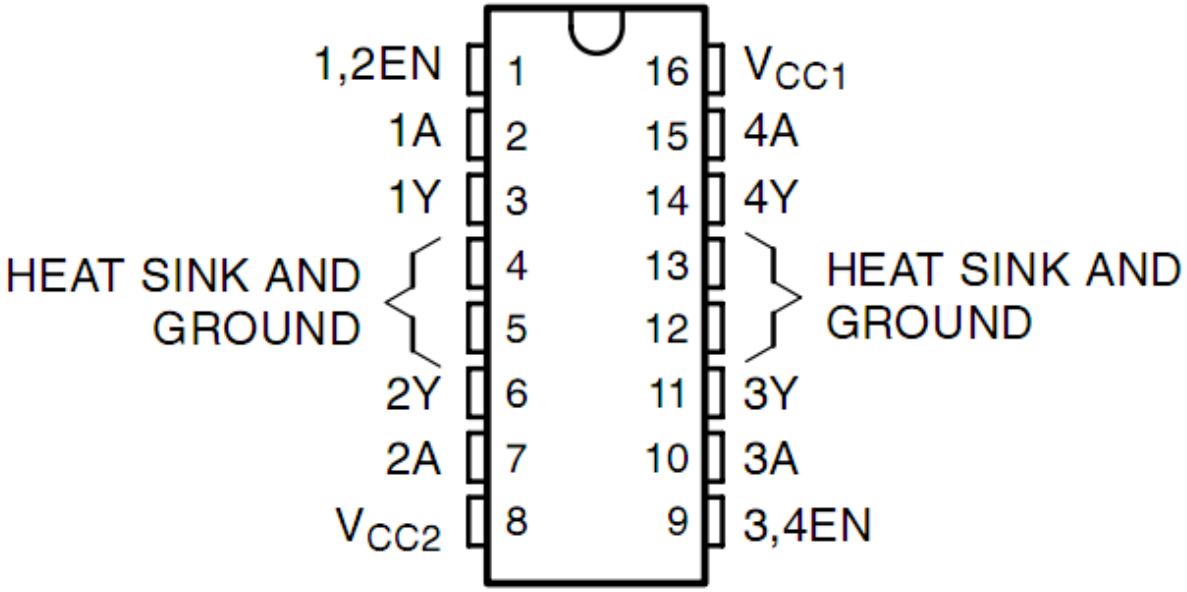
示例

- *第 18 课 74HC595* (Mega 板项目)
- *第 23 课简单创作 - 数字骰子* (Mega 板项目)
- *第 18 课 74HC595* （R3 板项目）
- *第 23 课简单创作 - 数字骰子* （R3 板项目）

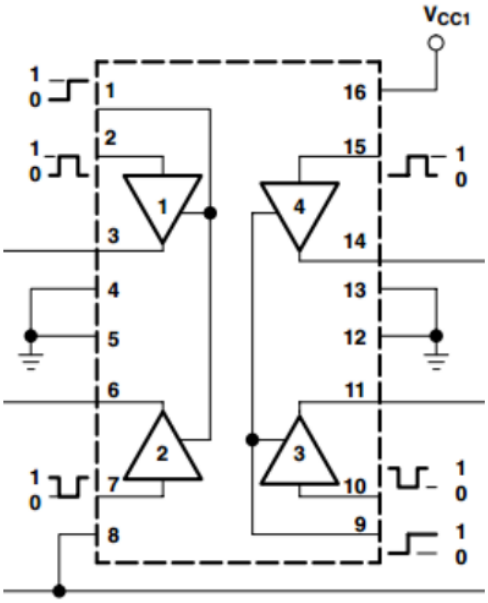
2.10 L293D

L293D 是一款高电压大电流芯片集成的 4 通道电机驱动器。它设计用于连接标准的 DTL、TTL 逻辑电平，驱动感性负载（如继电器线圈、直流、步进电机）和功率开关晶体管等。直流电机是将直流电能转化为机械能的装置。它们以其优越的调速性能被广泛应用于电力驱动。

请参见下面的引脚图。L293D 有两个引脚（Vcc1 和 Vcc2）用于供电。Vcc2 为电机供电，Vcc1 为芯片供电。由于此处使用小型直流电机，因此将两个引脚连接到 +5V。



以下是 L293D 的内部结构。EN 引脚为使能引脚，只工作在高电平；A 代表输入，Y 代表输出。你可以在右下角看到它们之间的关系。EN 为高电平时，A 为高电平时，Y 输出高电平；如果 A 为低电平，则 Y 输出低电平。当 EN 引脚为低电平时，L293D 不工作。



INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

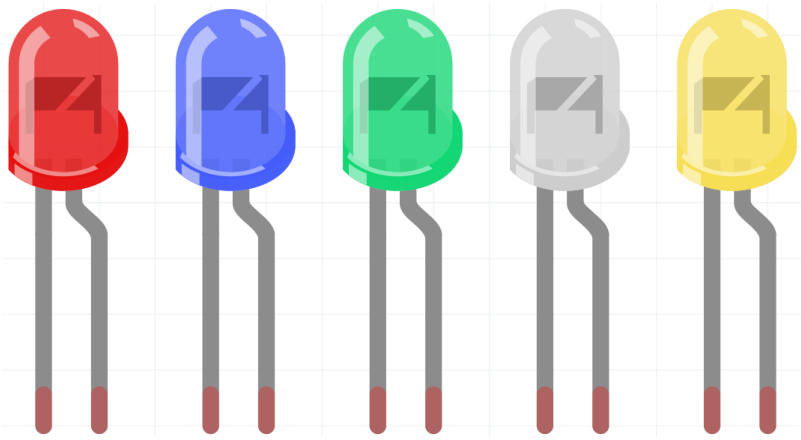
- L293D 数据表

例子

- *第 22 课简单创作 - 小风扇* (Mega 板项目)
- *第 22 课简单创作 - 小风扇* （R3 板项目）
- *13. 旋转风扇* (Scratch 项目)

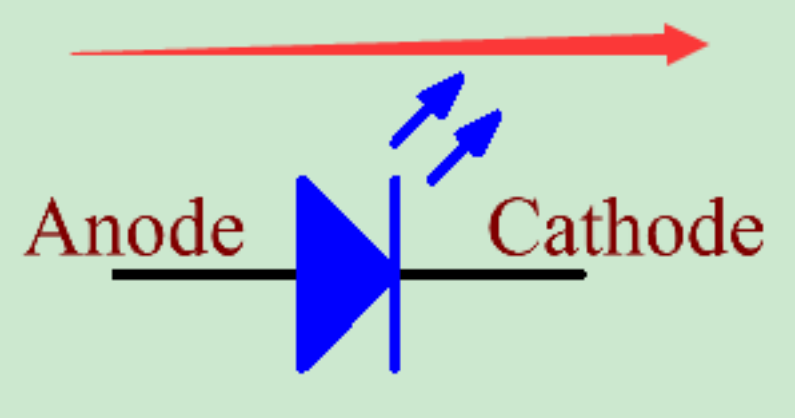
显示

2.11 LED 发光二极管



半导体发光二极管是一种可以通过 PN 结将电能转化为光能的元件。按波长可分为激光二极管、红外发光二极管和通常称为发光二极管（LED）的可见光发光二极管。

二极管具有单向导电性，因此电流将如图电路符号中的箭头所示。你只能为阳极提供正电源，为阴极提供负电源。因此 LED 将亮起。



LED 有两个引脚。较长的是阳极，较短的是阴极。注意不要接反。LED 有固定的正向压降，因此不能直接与电路连接，因为电源电压会超过这个压降而导致 LED 被烧毁。红黄绿 LED 的正向电压为 1.8V，白的为 2.6V，大多数 LED 最大能承受 20mA 的电流，所以需要串联一个限流电阻。

电阻值的计算公式如下：

R = (Vsupply –VD)/I

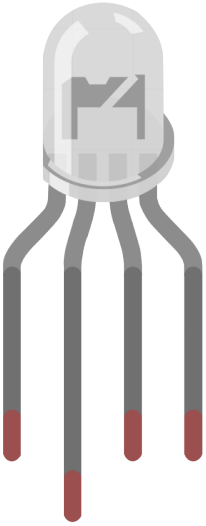
R 代表限流电阻的阻值，Vsupply 代表电压供应，VD 代表压降，I 代表 LED 的工作电流。

下面是 LED 的详细介绍：LED - 维基百科。

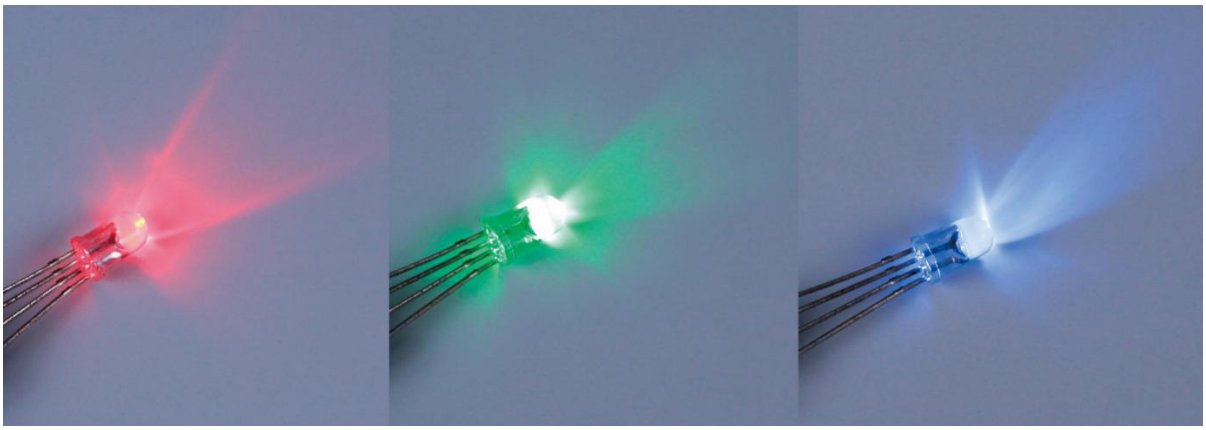
示例

- *第 1 课闪烁的 LED* (Mega 板项目)
- *第 8 课电位器* (Mega 板项目)
- *第 1 课闪烁的 LED* （R3 板项目）
- *第 8 课电位器* （R3 板项目）
- *2. 台灯* （Scratch 项目）
- *3. 呼吸灯* （Scratch 项目）

2.12 RGB LED

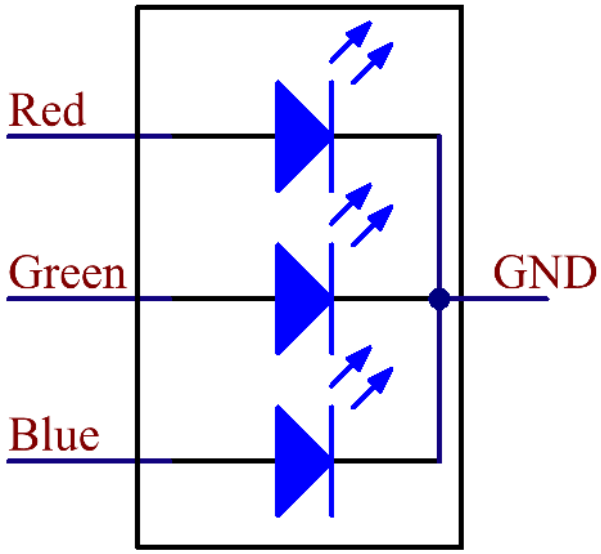


RGB LED 又名彩色发光二极管，它发出各种颜色的光。RGB LED 将红、绿、蓝三种 LED 封装在透明或半透明的塑料外壳中。它可以通过改变三个引脚的输入电压并叠加来显示各种颜色，据统计，可以产生 16,777,216 种不同的颜色。

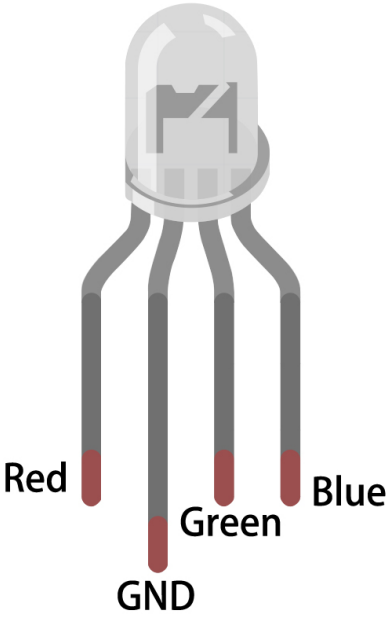


RGB LED 可分为共阳极和共阴极 LED。在本套件中，使用后者。在共阴极，或 CC，装置连接的三个 LED 的阴极。将其与 GND 连接并插入三个引脚后，LED 将闪烁相应的颜色。

其电路符号如图所示：



一个 RGB LED 有 4 个引脚：最长的一个是 GND；其他是红色、绿色和蓝色。触摸它的塑料外壳，你会发现一个切口。最靠近切口的引脚是第一个引脚，标记为红色，然后依次是 GND、绿色和蓝色。



示例

- *第 7 课 RGB LED* (Mega 板项目)
- *第 7 课 RGB LED* （R3 板项目）
- *4. 彩灯* (Scratch 项目)

2.13 7 段数码管

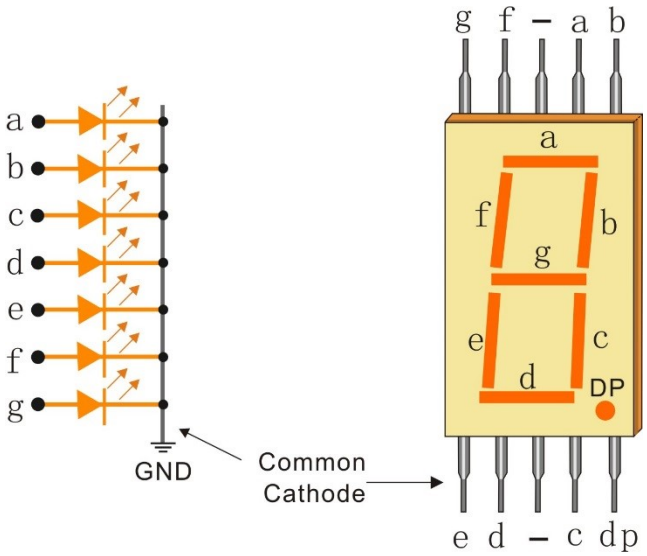
7 段数码管是一个 8 字形的组件，封装了 7 个 LED。每个 LED 称为一个段——当通电时，一个段构成要显示的数字（十进制和十六进制）的一部分。有时会在同一封装内使用额外的第 8 个 LED，因此当两个或多个 7 段数码管连接在一起以显示大于 10 的数字时，可以指示小数点 (DP)。

数码管中的每个 LED 都有一个位置段，其中一个连接引脚从矩形塑料封装中引出。这些 LED 引脚从“a”到“g”标记，代表每个单独的 LED。其他 LED 引脚连接在一起形成一个公共引脚。因此，通过按特定顺序正向偏置 LED 段的适当引脚，某些段会变亮而其他段会保持暗淡，从而在显示屏上显示相应的字符。

数码管的常用引脚一般说明其类型。引脚连接有两种：一种是接阴极的引脚，一种是接阳极的引脚，分别表示共阴极（CC）和共阳极（CA）。顾名思义，当 CA 数码管连接了 7 个段的所有阳极时，CC 数码管连接了 7 个 LED 的所有阴极。connected.

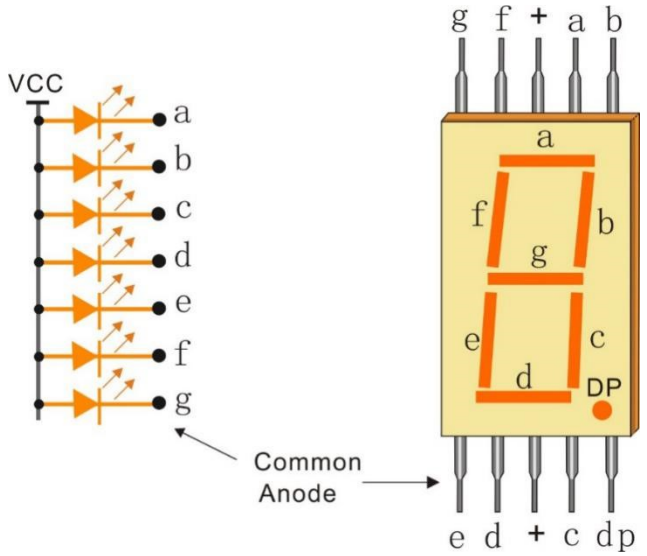
共阴极 7 段数码管

在共阴极数码管中，所有 LED 段的阴极都连接到逻辑“0”或接地。然后一个单独的段 (a-imgg) 被一个“高”或逻辑“1”信号通电，通过限流电阻正向偏置该段的阳极。



共阳极 7 段数码管

在共阳极数码管中，所有 LED 段的阳极都连接到逻辑“1”。然后单个段 (ag) 由接地、逻辑“0”或“低”信号通电，通过限流电阻连接到该段的阴极。



显示代码

为了帮助你了解 7 段显示器（共阴极）如何显示数字，我们绘制了下表。数字是 7 段显示器上显示的数字 0-F；(DP) GFEDCBA 是指对应的 LED 设置为 0 或 1，例如 00111111 表示 DP 和 G 设置为 0，其他则设置为 1。因此，在 7 段显示器上显示数字 0，而 HEX Code 对应的是十六进制数。

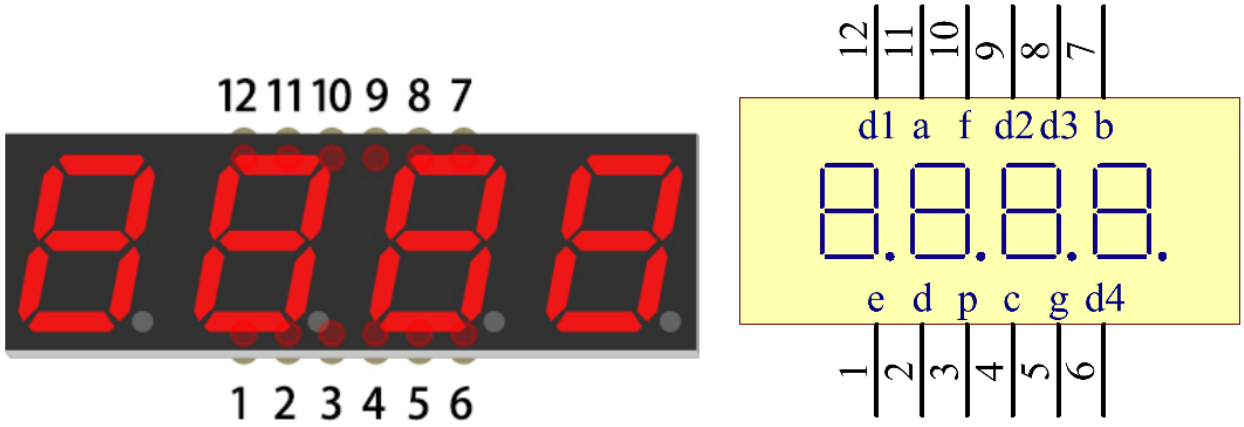
Numbers	Common Cathode		Numbers	Common Cathode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA	Hex Code
0	00111111	0x3f	A	01110111	0x77
1	00000110	0x06	B	01111100	0x7c
2	01011011	0x5b	C	00111001	0x39
3	01001111	0x4f	D	01011110	0x5e
4	01100110	0x66	E	01111001	0x79
5	01101101	0x6d	F	01110001	0x71
6	01111101	0x7d			
7	00000111	0x07			
8	01111111	0x7f			
9	01101111	0x6f			

示例

- *第 17 课 7 段数码管* (Mega 板项目)
- *第 23 课简单创作 - 数字骰子* (Mega 板项目)
- *第 17 课 7 段数码管* （R3 板项目）
- *第 23 课简单创作 - 数字骰子* （R3 板项目）

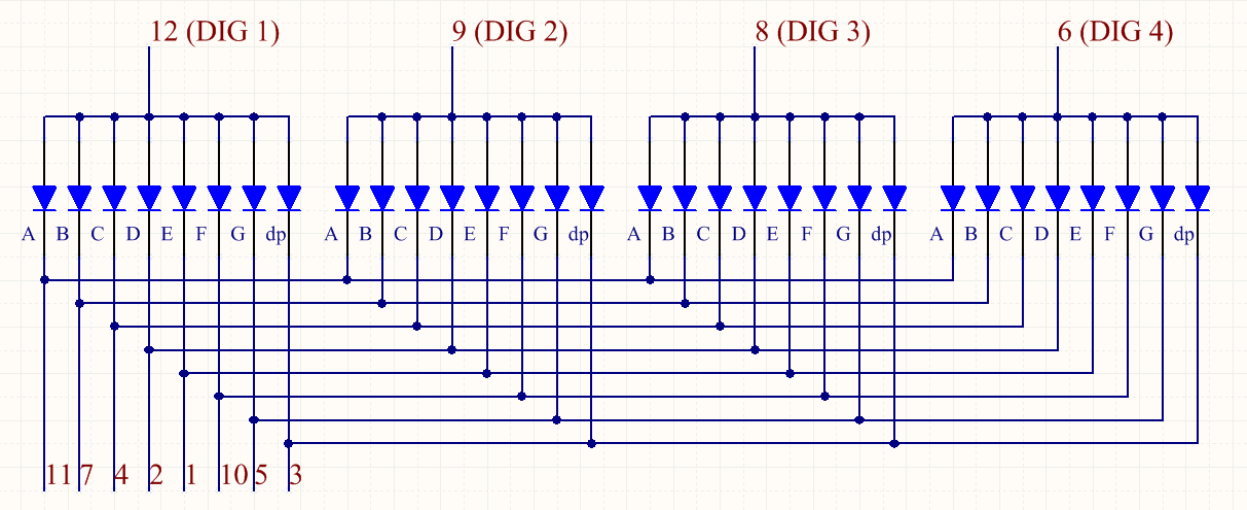
2.14 4 位 7 段数码管

4 位 7 段数码管由四个协同工作的 7 段数码管组成。



4 位 7 段数码管独立工作。它利用人类视觉暂留的原理，将每个 7 段的字符快速循环显示，形成连续的字符串。

例如，当数码管显示 “1234” 时，第一个 7 段显示 “1”，不显示 “234”。一段时间后，第 2 个 7 段显示 “2”，第 1 个第 3 个第 4 个 7 段不显示，以此类推，四位数码管依次显示。这个过程很短（一般为 5ms），而且由于光学余辉效应和视觉残留原理，我们可以同时看到四个字符。



显示代码

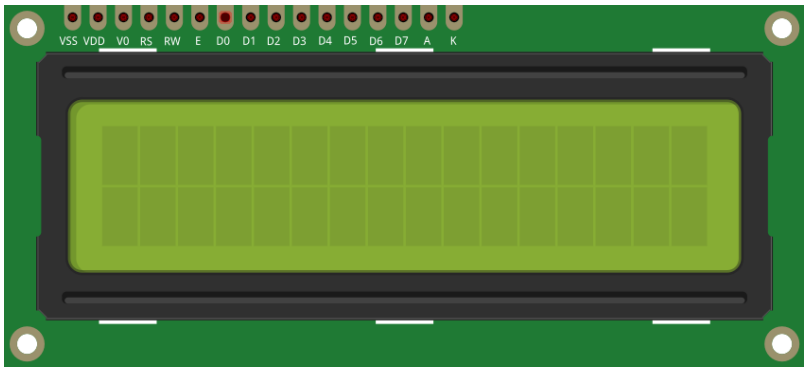
为了帮助你了解 7 段式数码管（共阳极）如何显示数字，我们绘制了下表。数字是 7 段数码管上显示的数字 0-F；(DP) GFEDCBA 是指对应的 LED 设置为 0 或 1，例如 11000000 表示 DP 和 G 设置为 1，其他则设置为 0，因此在 7 段数码管上显示数字 0，而 HEX Code 对应的是十六进制数。

Numbers	Common Anode		Numbers	Common Anode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA	Hex Code
0	11000000	0xc0	A	10001000	0x88
1	11111001	0xf9	B	10000011	0x83
2	10100100	0xa4	C	11000110	0xc6
3	10110000	0xb0	D	10100001	0xa1
4	10011001	0x99	E	10000110	0x86
5	10010010	0x92	F	10001110	0x8e
6	10000010	0x82	.	01111111	0x7f
7	11111000	0xf8			
8	10000000	0x80			
9	10010000	0x90			

示例

- *第 20 课简单创作 - 秒表* (Mega 板项目)
- *第 20 课简单创作 - 秒表* （R3 板项目）

2.15 LCD1602 液晶显示屏



LCD1602，即 1602 字符型液晶显示器，是一种显示字母、数字、字符等的点阵模块。它由 5x7 或 5x11 点阵位置组成；每个位置可以显示一个字符。两个字符之间有一个点间距，行之间有一个空格，从而将字符和行分开。数字 1602 表示在显示屏上可以显示 2 行，每行 16 个字符。现在让我们检查更多细节！

LCD1602 一般都有并口，即同时控制几个管脚。LCD1602 可分为八端口和四端口连接。如果使用八端口连接，那么控制板的所有数字端口几乎都被占用了。如果要连接更多传感器，将没有可用端口。因此，这里采用四端口连接，以便更好地应用。

引脚介绍

- **VSS:** 接地
- **VDD:** 接 +5V 电源
- **VO:** 调整对比度
- **RS:** 寄存器选择引脚，用于控制你将数据写入 LCD 存储器中的哪个位置。你可以选择数据寄存器（保存屏幕上的内容）或指令寄存器（LCD 的控制器在此处查找有关下一步操作的指令）。
- **R/W:** 一个读/写引脚，用于在读和写模式之间进行选择。
- **E:** 接收到高电平 (1) 时读取信息的使能引脚。当信号从高电平变为低电平时运行指令。
- **D0-D7:** 读写数据
- **A/K:** 控制 LCD 背光的引脚。将 K 连接到 GND，将 A 连接到 3.3v。打开背光灯，在比较暗的环境中会看到清晰的字符。

示例

- *第 11 课 LCD1602* (Mega 板项目)
- *第 15 课温湿度传感器* (Mega 板项目)
- *第 11 课 LCD1602* （R3 板项目）
- *第 15 课温湿度传感器* （R3 板项目）
- *5. 跳动的字符* （Scratch 项目）

声音

2.16 蜂鸣器

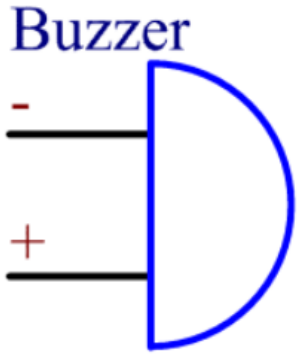
蜂鸣器作为一种一体化结构的电子蜂鸣器，由直流电源供电，广泛应用于计算机、打印机、复印机、报警器、电子玩具、汽车电子设备、电话、定时器等电子产品或语音设备中。

蜂鸣器可分为有源和无源蜂鸣器。转动蜂鸣器，使其引脚朝上，带有绿色电路板的蜂鸣器为无源蜂鸣器，用黑色胶带封住的为有源蜂鸣器。

有源蜂鸣器和无源蜂鸣器的区别：

有源蜂鸣器内置振荡源，通电时会发出声音。但是无源蜂鸣器没有这样的源，所以如果使用直流信号，它不会发出哔哔声；相反，你需要使用频率在 2K 到 5K 之间的方波来驱动它。由于有多个内置振荡电路，有源蜂鸣器通常比无源蜂鸣器贵。

以下是蜂鸣器的电气符号。它有两个引脚，正极和负极。表面有一个 + 代表阳极，另一个是阴极。



你可以检查一下蜂鸣器的引脚，长的是阳极，短的是阴极。连接时请不要混淆，否则蜂鸣器不会发出声音。

蜂鸣器 - 维基百科

示例

- 第 *4* 课蜂鸣器 (Mega 板项目)
- 第 *21* 课简单创作 - 抢答器 (Mega 板项目)
- 第 *4* 课蜂鸣器 （R3 板项目）
- 第 *21* 课简单创作 - 抢答器 （R3 板项目）

执行器

2.17 直流电机

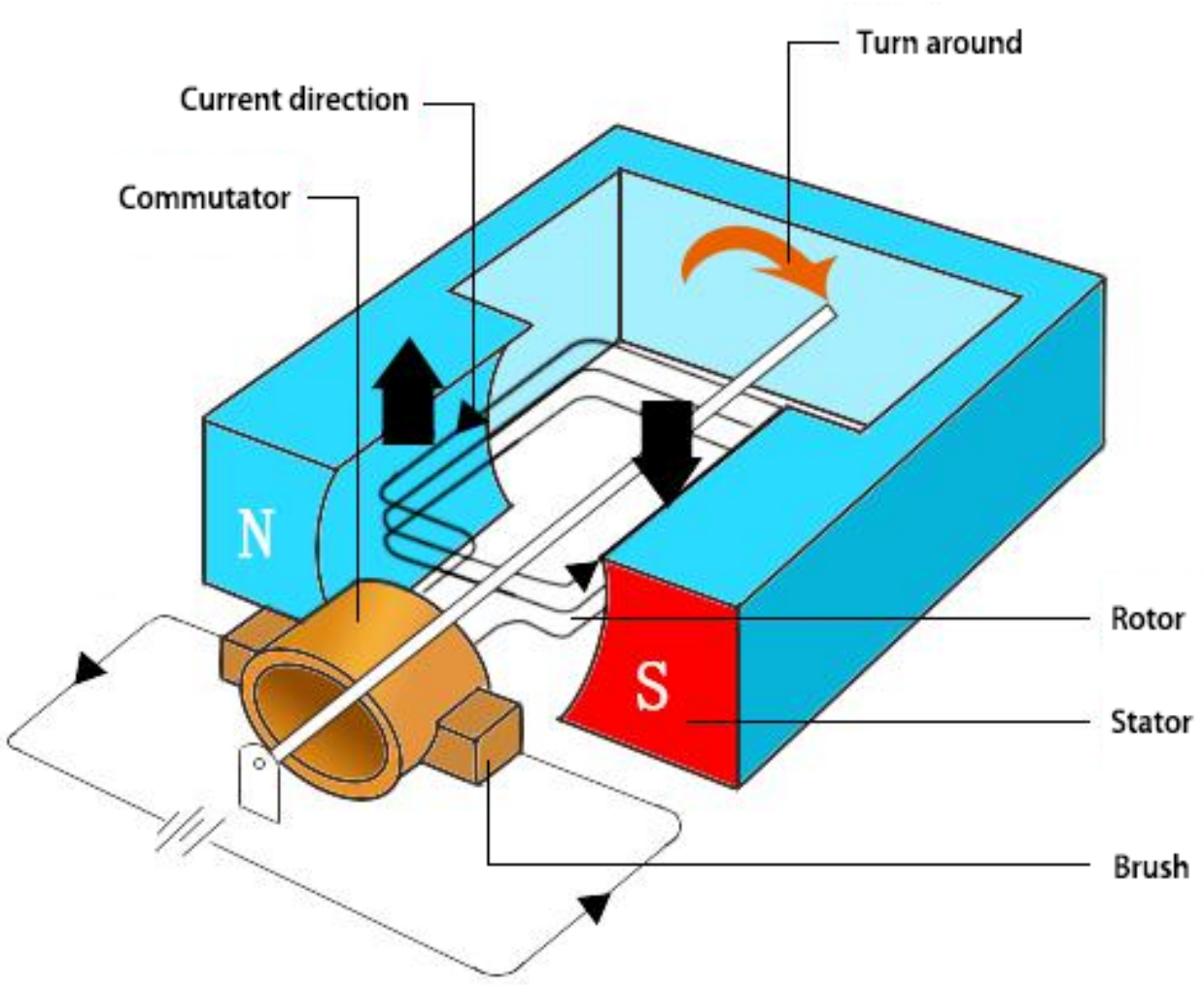
这是一个 3V 直流电机。当你给 2 个端子中的每个端子一个高电平和一个低电平时，它会旋转。

- **尺寸**: 25*20*15MM
- **工作电压**: 1-6V
- **空转电流** (3V): 70m
- **空转速度** (3V): 13000RPM

- **堵转电流** (3V): 800mA
- **轴径**: 2mm

直流 (DC) 电机是一种连续执行器，可将电能转换为机械能。直流电机通过产生连续的角旋转使旋转泵、风扇、压缩机、叶轮和其他设备工作。

直流电机由两部分组成，电机的固定部分称为定子，电机的内部部分称为转子（或直流电机的电枢），通过旋转产生运动。产生运动的关键是将电枢放置在永磁体的磁场内（其磁场从北极延伸到南极）。磁场和移动的带电粒子（载流导线产生磁场）的相互作用产生旋转电枢的扭矩。



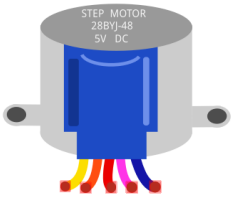
电流从电池的正极流过电路，通过铜刷流向换向器，然后流向电枢。但是由于换向器中的两个间隙，该流动在每次完整旋转中途反转。这种连续的反转实质上是将来自电池的直流电转换为交流电，使电枢在正确的时间在正确的方向上经历扭矩以保持旋转。

- 直流电机 - MagLab

示例

- *第 22 课简单创作 - 小风扇* (Mega 板项目)
- *第 22 课简单创作 - 小风扇* （R3 板项目）
- *13. 旋转风扇* (Scratch 项目)

2.18 步进电机



步进电机由于其独特的设计，可以在没有任何反馈机制的情况下进行高精度控制。步进电机的轴上装有一系列磁铁，由一系列电磁线圈控制，这些线圈按特定顺序带正负电，以小“步”精确地向前或向后移动。

原理

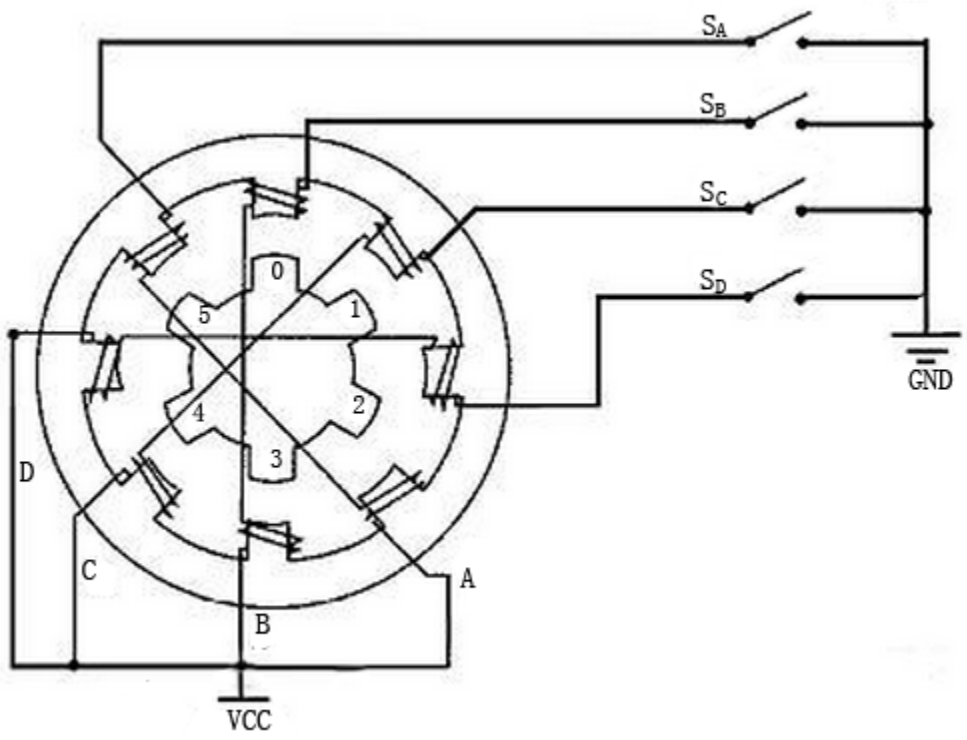
步进电机有两种类型，单极和双极，了解你使用的是哪种类型非常重要。在本实验中，我们将使用单极步进电机。

步进电机是四相电机，使用单极性直流电源。只要按适当的时序给电机的所有相绕组通电，就可以使其逐步转动。四相无功步进电机原理图：

图中，电机的中间是一个转子——一个齿轮状的永磁体。在转子周围，0 到 5 是齿。再往外，有 8 个磁极，每两个相对的磁极通过线圈绕组相连。所以它们从 A 到 D 形成四对，称为相。它有四根引线连接开关 SA、SB、SC 和 SD。因此，电路中四相并联，一相的两个磁极串联。

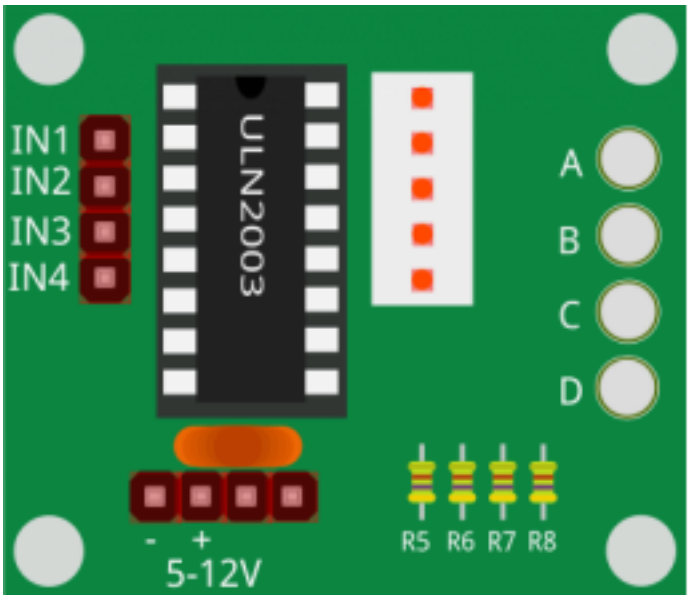
以下是 4 相步进电机的工作原理：

开始时，开关 SB 通电，开关 SA、SC、SD 断电，B 相磁极对准转子的 0、3 齿。同时，齿 1 和齿 4 产生 C 相和 D 相极的交错齿。齿 2 和 5 生成具有 D 和 A 相极的交错齿。当开关 SC 通电时，开关 SB、SA、SD 断电，转子在 C 相绕组的磁场和 1、4 齿之间的磁场作用下旋转。此时 1、4 齿与 C-的磁极对齐相绕组。而 0 号和 3 号齿产生 A、B 相磁极的交错齿，2、5 号齿产生 A、D 相磁极的交错齿。类似的情况还在继续。依次给 A、B、C、D 相通电，转子按 A、B、C、D 的顺序转动。



四相步进电机有单四步、双四步、八步三种运行方式。单四步和双四步的步距角相同，但单四步的驱动力矩较小。八步的步距角是单四步和双四步的一半。因此，八步工作模式可以保持较高的驱动扭矩，提高控制精度。在这个实验中，我们让步进电机工作在八步模式。

ULN2003 模块



要将电机应用到电路中，需要使用驱动板。步进电机驱动器- ULN2003 (数据表) 是一个 7 通道逆变器电路。即当输入端为高电平时，ULN2003 的输出端为低电平，反之亦然。如果我们给 IN1 提供高电平，给 IN2、IN3 和 IN4 提供低电平，那么输出端 OUT1 为低电平，其他所有输出端都为高电平。于是 D1 亮，开关 SA 通电，步进电机转一圈。类似的情况不断重复。因此，只要给步进电机一个特定的时序，它就会一步一步地转动。

这里的 ULN2003 用于为步进电机提供特定的时序。

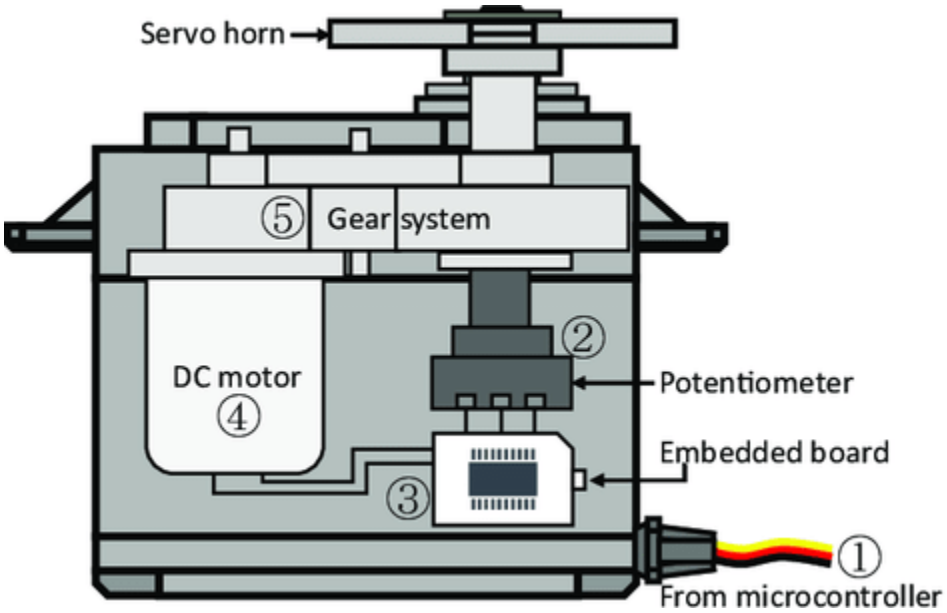
示例

- 第 *19* 课步进电机 (Mega 板项目)
- 第 *19* 课步进电机 （R3 板项目）

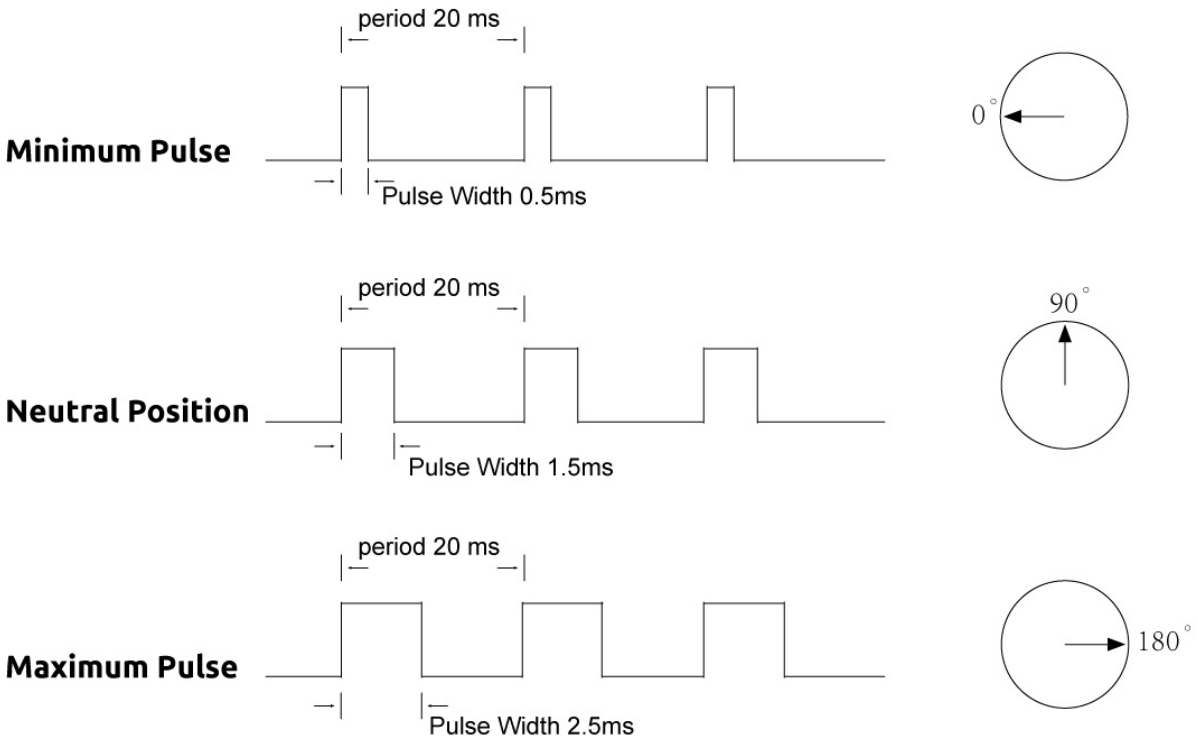
2.19 舵机

一个舵机一般由以下几部分组成：外壳、轴（Servo horn）、齿轮系统（Gear system）、电位器 (Potentiometer)、直流电机 (DC Motor) 和嵌入式板 (Embedded board)。

其工作原理是：微控制器向舵机发出 PWM 信号，然后舵机中的嵌入式板卡通过信号引脚接收信号并控制里面的电机转动。结果，电机驱动齿轮系统，然后在减速后驱动轴。舵机的轴和电位器连接在一起。当轴旋转时，它驱动电位器，因此电位器向嵌入式板输出电压信号。然后棋盘根据当前位置确定旋转的方向和速度，因此它可以准确地停在定义的正确位置并保持在那里。



该角度由施加到控制线的脉冲持续时间决定。这称为脉宽调制。伺服器期望每 20 毫秒看到一个脉冲。脉冲的长度将决定电机转动的距离。例如，一个 1.5ms 的脉冲将使电机转动到 90 度位置（中立位置）。当一个小于 1.5 ms 的脉冲发送到伺服系统时，伺服系统会旋转到一个位置，并将其输出轴从中性点逆时针方向保持一定度数。当脉冲宽度超过 1.5 ms 时，情况正好相反。将命令舵机转向有效位置的最小脉冲宽度和最大脉冲宽度是每个舵机的功能。通常，最小脉冲宽度约为 0.5 毫秒，最大脉冲宽度约为 2.5 毫秒。



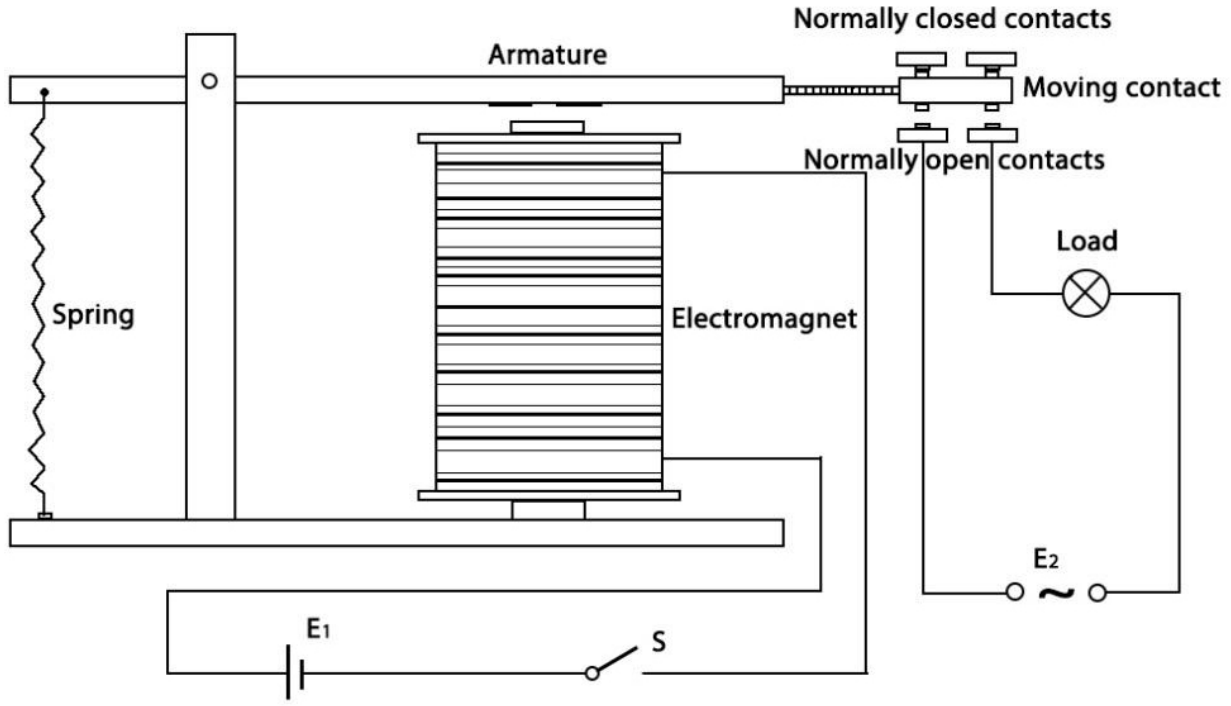
示例

- 第 *10* 课舵机 (Mega 板项目)
- 第 *10* 课舵机 （R3 板项目）
- *12.* 摆钟 (Scratch 项目)

2.20 继电器

我们可能知道，继电器是一种设备，用于响应所施加的输入信号在两个或多个点或设备之间提供连接。换句话说，继电器在控制器和设备之间提供隔离，因为设备可以在交流电和直流电下工作。然而，它们从工作于直流的微控制器接收信号，因此需要继电器来弥补差距。当你需要用小电信号控制大量电流或电压时，继电器非常有用。

每个继电器有 5 个部分：



Electromagnet - 它由一个由线圈缠绕的铁芯组成。当电流通过时，它会变成磁性的。因此，它被称为电磁铁。

Armature - 可移动的磁条称为电枢。当电流流过它们时，线圈会通电，从而产生磁场，用于接通或断开常开 (N/O) 或常闭 (N/C) 点。并且电枢可以用直流电 (DC) 和交流电 (AC) 移动。

Spring - 当没有电流流过电磁铁上的线圈时，弹簧会将衔铁拉开，因此电路无法完成。

Normally open/closed contacts- 有两个触点：

- 常开 - 继电器激活时连接，不活动时断开。
- 常闭 - 继电器激活时不连接，不活动时连接。

Molded frame- 继电器覆盖有塑料以提供保护。

继电器的工作原理很简单。当继电器通电时，电流开始流过控制线圈；结果，电磁铁开始通电。然后衔铁被吸引到线圈上，将动触点一起拉下，从而与常开触点连接。因此带负载的电路通电。然后断开电路会发生类似的情况，因为动触点将在弹簧力的作用下被拉到常闭触点。这样，继电器的通断可以控制负载电路的状态。

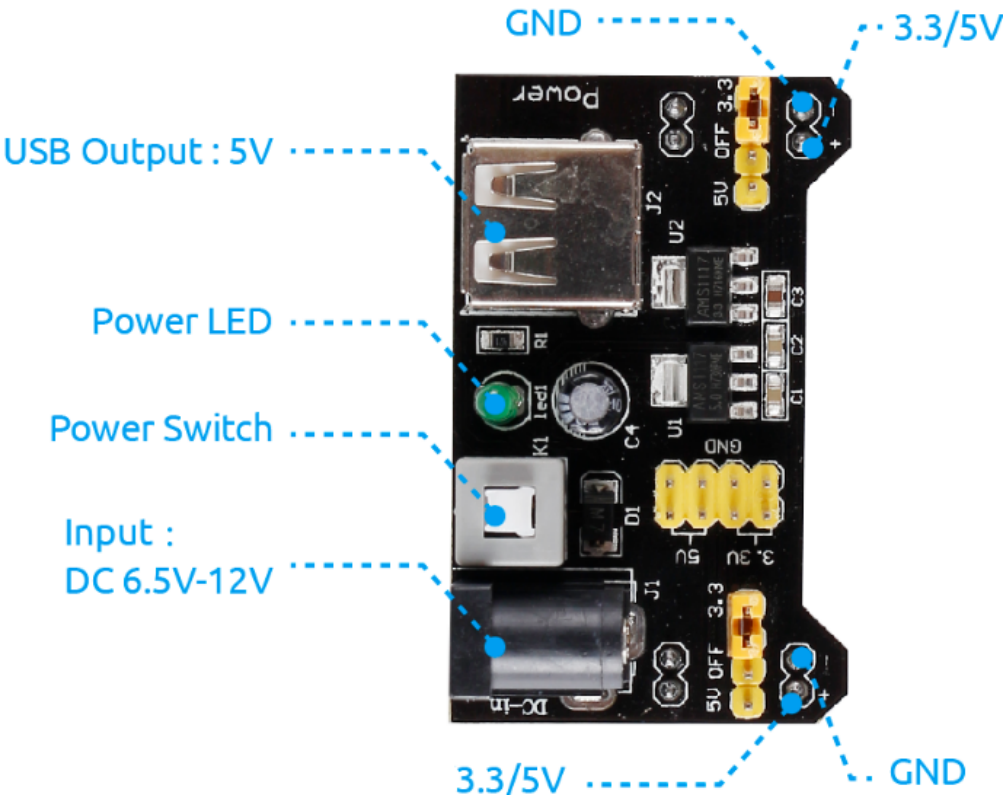
示例

- 第 6 课继电器 (Mega 板项目)
- 第 6 课继电器 （R3 板项目）

2.21 电源模块

当我们需要大电流来驱动某个组件时，会严重干扰主板的正常工作。因此，我们通过该模块单独为组件供电，使其运行安全稳定。

你只需将其插入面包板即可供电。它提供 3.3V 和 5V 的电压，你可以通过随附的跳线帽进行连接。

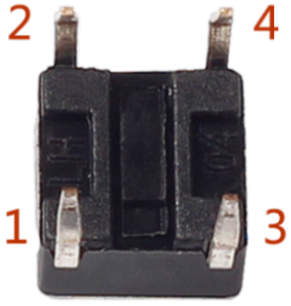


特性和规格

- 输入电压：6.5 - 12V
- 两个独立通道
- 输出电压：5V、3.3V（可通过跳线帽调节，0V、3.3V 和 5V 配置）
- 输出电流：最大输出电流 700mA
- 板载公头排针，用于 GND、5V、3.3V 输出
- ON-OFF 开关可用。
- USB (Type-A) 输入可用。
- 直流公头排针输入可用。
- 板载电源 LED
- 尺寸：53mm x 33mm（长 x 宽）

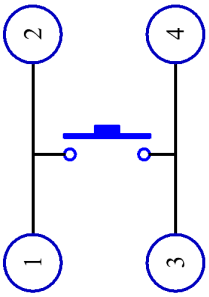
控制器

2.22 按键

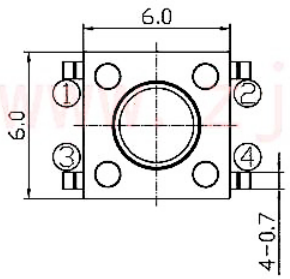


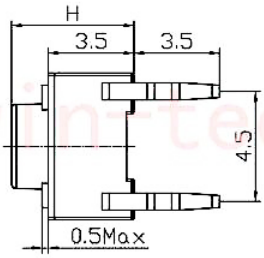
按键是用于控制电子设备的常见组件，它们通常用作连接或断开电路的开关。

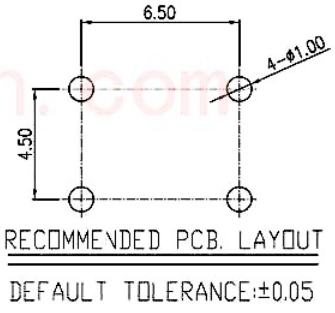
下面是一个按钮的内部结构。右下方的符号通常用于表示电路中的按钮。

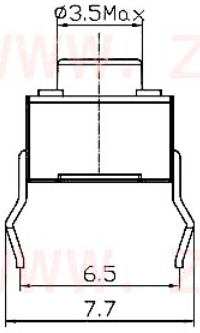


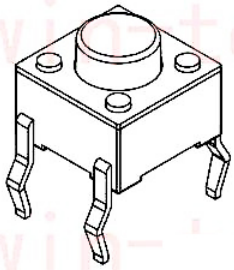
由于引脚 1 连接到引脚 2，引脚 3 连接到引脚 4，当按下按键时，4 个引脚连接，从而关闭电路。

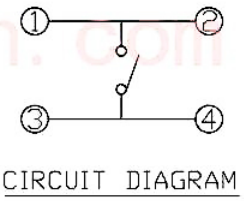








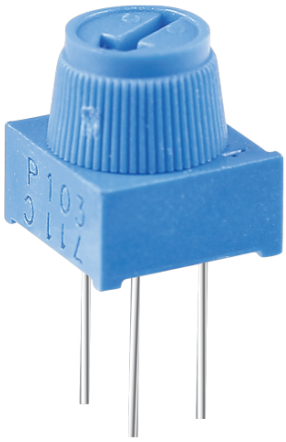




示例

- *第 3 课按键* (Mega 板项目)
- *第 21 课简单创作 - 抢答器* (Mega 板项目)
- *第 3 课按键* （R3 板项目）
- *第 21 课简单创作 - 抢答器* （R3 板项目）
- *7. 门铃* （Scratch 项目）
- *15. 游戏 - 吃苹果* （Scratch 项目）

2.23 电位器

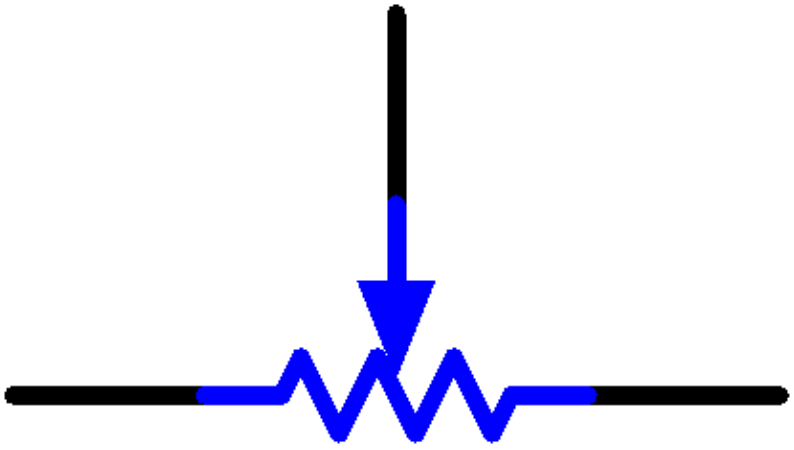


电位器也是一个有 3 个端子的电阻元件，它的电阻值可以根据一些规律的变化进行调整。

电位器有各种形状、大小和值，但它们都有以下共同点：

- 它们具有三个端子（或连接点）。
- 它们有一个旋钮、螺丝或滑块，可以移动以改变中间端子和任一外部端子之间的电阻。
- 随着旋钮、螺钉或滑块的移动，中间端子和任一外部端子之间的电阻从 0 Ω 到电位器的最大电阻变化。

这是电位器的电路符号：



电位器在电路中的作用如下：

用作分压器

电位器是一个连续可调的电阻器。当你调整电位器的转轴或滑动手柄时，活动触点会在电阻上滑动。此时，可以根据施加在电位器上的电压和可动臂旋转的角度或移动的距离来输出电压。

用作变阻器

当电位器用作变阻器时，将中间引脚与电路中的其他 2 个引脚之一连接。因此，你可以在动触点的行程内获得平滑且连续变化的电阻值。

作为电流控制器

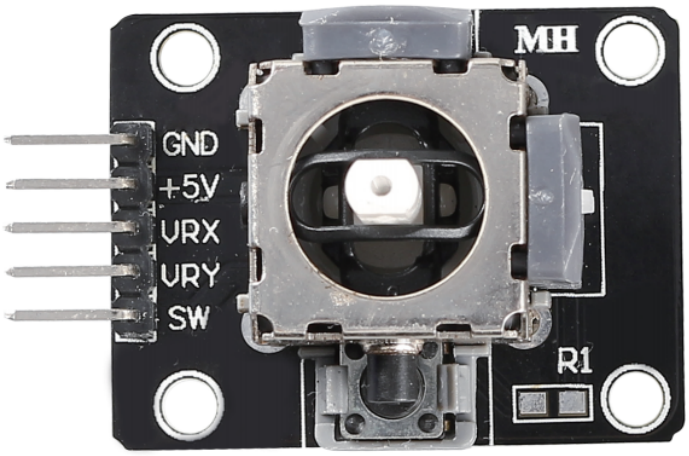
当电位器作为电流控制器时，必须连接滑触端子作为输出端子之一。

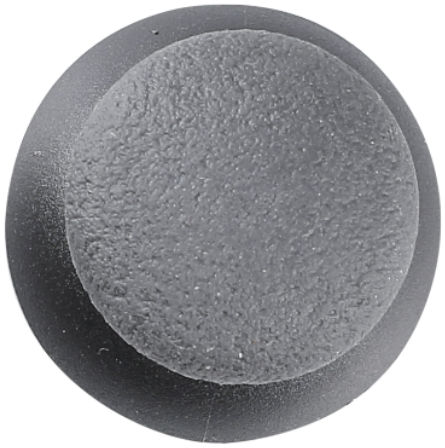
如果你想了解更多关于电位器的信息，请参考：电位器 - 维基百科.

示例

- 第 9 课光敏电阻 (Mega 板项目)
- 第 9 课光敏电阻 （R3 板项目）
- 6. 移动的老鼠 (Scratch 项目)
- 17. 游戏 - 打砖块 (Scratch 项目)

2.24 摇杆模块



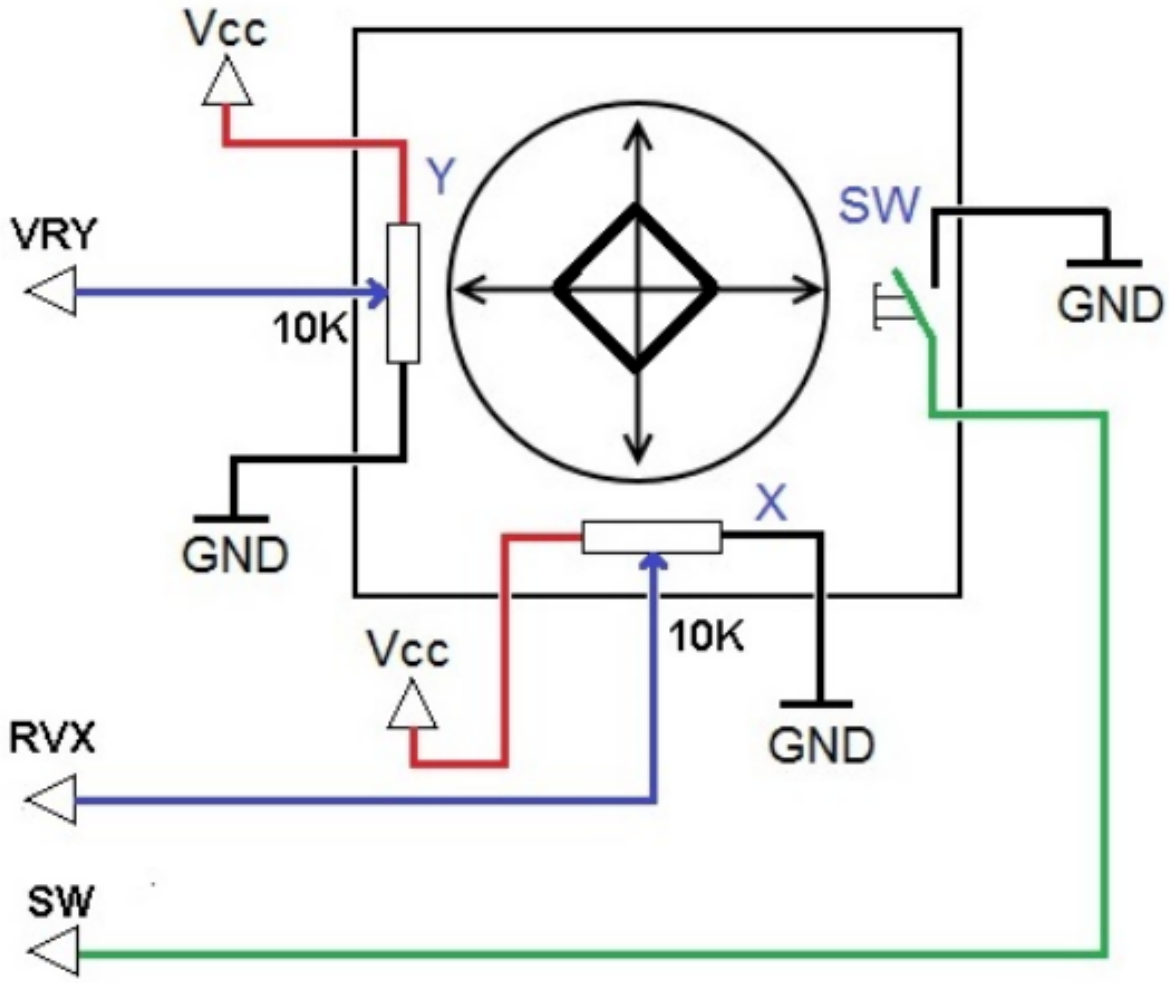


摇杆的基本思想是将摇杆的运动转化为计算机可以处理的电子信息。

为了将完整的运动范围传达给计算机，摇杆需要在两个轴上测量摇杆的位置——X 轴（从左到右）和 Y 轴（上下）。就像在基本几何中一样，XY 坐标精确地确定了摇杆的位置。

为了确定摇杆的位置，摇杆控制系统只需监控每个轴的位置。传统的模拟摇杆设计使用两个电位计或可变电阻器来实现这一点。

摇杆还有一个数字输入，当按下摇杆时该输入被激活。

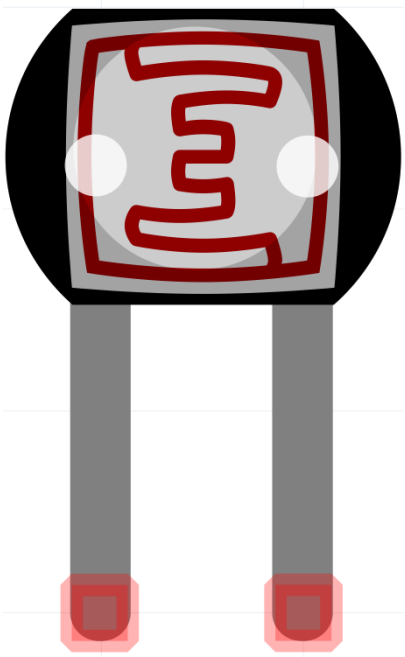


示例

- 第 *16* 课摇杆 (Mega 板项目)
- 第 *16* 课摇杆 （R3 板项目）
- *14.* 游戏 - 星际穿越 (Scratch 项目)
- *19.* 游戏 - 击败恶龙 (Scratch 项目)

传感器

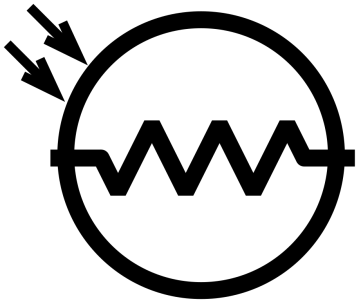
2.25 光敏电阻



光敏电阻或光电管是一种光控可变电阻器。光敏电阻的电阻随着入射光强度的增加而降低；换句话说，它表现出光导性。

光敏电阻器可应用于光敏检测器电路和光激活和暗激活开关电路中作为电阻半导体。在黑暗中，光敏电阻的电阻可高达几兆欧 (MΩ)，而在光照下，光敏电阻的电阻可低至几百欧姆。

这里是光敏电阻的电子符号。

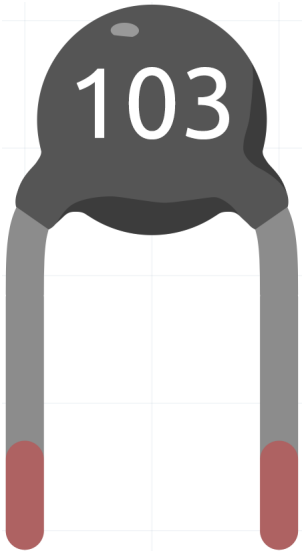


- 光敏电阻 - 维基百科

示例

- 第 9 课光敏电阻 (Mega 板项目)
- 第 9 课光敏电阻 （R3 板项目）
- *10.* 光控闹钟 (Scratch 项目)

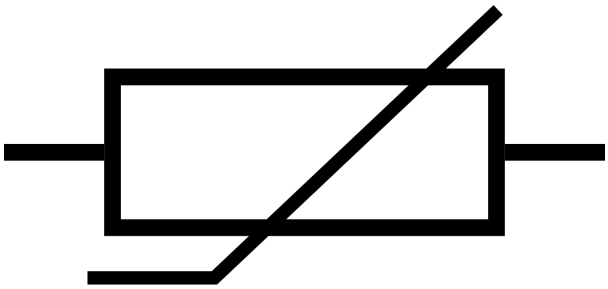
2.26 热敏电阻



热敏电阻是一种电阻器，其电阻强烈地依赖于温度，比标准电阻器的电阻更重要。这个词是热电阻和电阻的组合。热敏电阻广泛用作浪涌电流限制器、温度传感器（通常为负温度系数或 NTC 类型）、自复位过流保护器和自调节加热元件（通常为正温度系数或 PTC 类型）。

- 热敏电阻 - 维基百科

这是热敏电阻的电子符号：



热敏电阻有两种相反的基本类型：

- 对于 NTC 热敏电阻，电阻会随着温度升高而降低，这通常是由于价带中的热搅动引起的传导电子增加。NTC 通常用作温度传感器，或与电路串联用作浪涌电流限制器。
- 对于 PTC 热敏电阻，电阻会随着温度升高而增加，这通常是由于热晶格搅动增加，尤其是杂质和缺陷引起的热晶格搅动增加。PTC 热敏电阻通常与电路串联安装，用作自恢复保险丝以防止过流情况。

在此套件中，我们使用 NTC 套件。每个热敏电阻都有一个正常的电阻。这里是 10k ohm，这是在 25 摄氏度下测量的。

这是电阻和温度之间的关系：

RT = RN * expB(1/TK –1/TN)

- **RT** 是温度为 TK 时 NTC 热敏电阻的阻值。
- **RN** 为 NTC 热敏电阻在额定温度 TN 下的阻值。这里，RN 的数值是 10k。
- **TK** 是开尔文温度，单位是 K。这里 TK 的数值是 273.15+ 摄氏度。

- **TN** 是额定开尔文温度；单位也是 K。这里，TN 的数值是 273.15+25。
- 而 **B(beta)**，NTC 热敏电阻的材料常数，也称为热敏指数，数值为 3950。
- **exp** 是 exponential 的缩写，基数 e 为自然数，约等于 2.7。

转换这个公式 TK=1/(ln(RT/RN)/B+1/TN) 得到开尔文温度，负 273.15 等于摄氏度。

这个关系是一个经验公式。只有当温度和电阻在有效范围内时才准确。

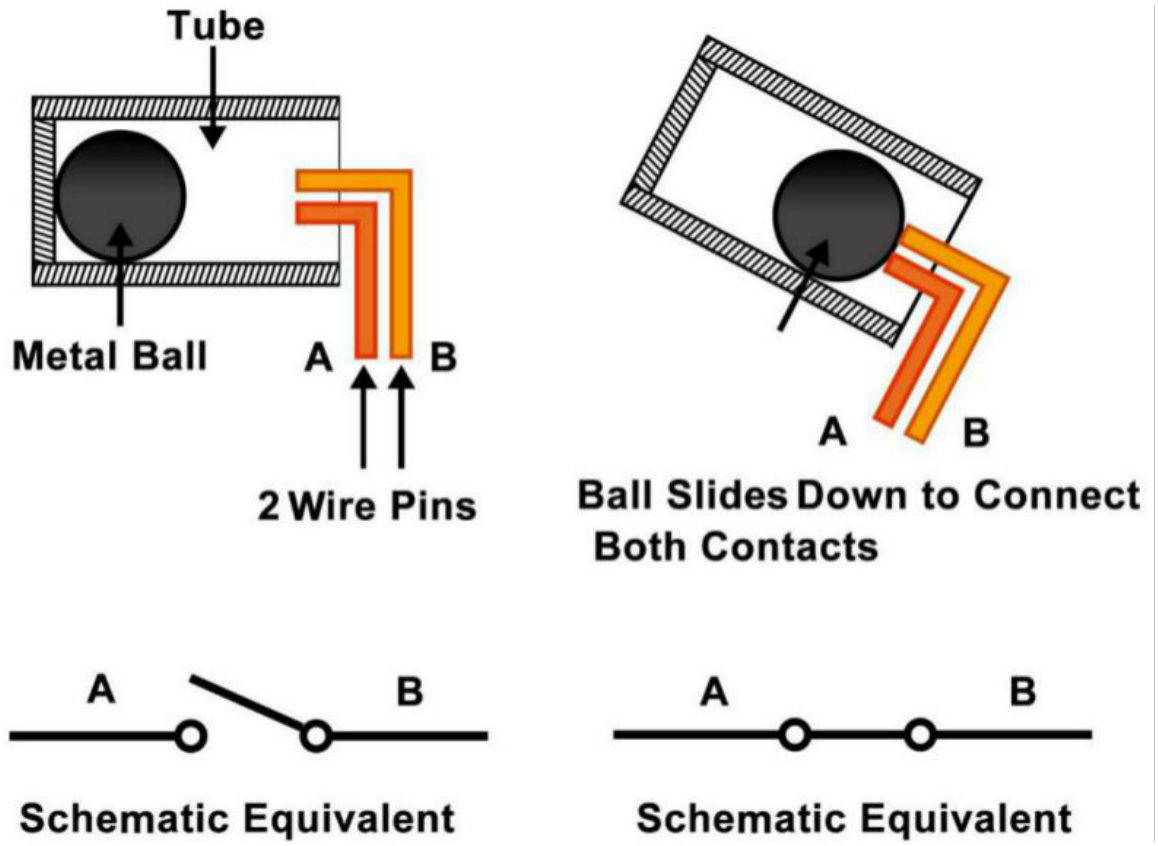
示例

- *第 12 课热敏电阻* (Mega 板项目)
- *第 12 课热敏电阻* （R3 板项目）
- *9. 低温报警器* (Scratch 项目)

2.27 倾斜开关

这里使用的倾斜开关是里面有一个金属球。它用于检测小角度的倾斜度。

原理很简单。当开关倾斜 一定角度时，里面的小球滚下来，接触到外面的引脚所连接的两个触点，从而触发电路。否则球将远离触点，从而断开电路。



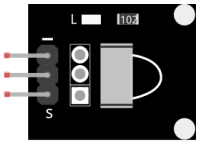
- SW520D 倾斜开关数据表

示例

- 第 5 课倾斜开关 (Mega 板项目)
- 第 5 课倾斜开关 （R3 板项目）
- 8. 不倒翁 (Scratch 项目)

2.28 红外接收模块

红外接收模块



- S：信号输出
- +：VCC
- -: 地

红外接收器是接收红外信号并能独立接收红外线并输出兼容 TTL 电平的信号的部件。它的尺寸与普通的塑料封装晶体管相似，适用于各种红外遥控和红外传输。

红外线（IR）通信是一种流行的、低成本的、易于使用的无线通信技术。红外光的波长比可见光稍长，因此人眼无法察觉——非常适合无线通信。红外通信常用的调制方案是 38KHz 调制。

- 采用 HX1838 红外接收传感器，灵敏度高
- 可用于远程控制
- 电源：5V
- 接口：数字
- 调制频率：38Khz
- 引脚定义：(1) 输出 (2) Vcc (3) GND
- 尺寸：23.5mm x 21.5mm

遥控器

这是一款迷你薄型红外无线遥控器，具有 21 个功能按钮，传输距离可达 8 米，适合在儿童房操作各种设备。

- 尺寸：85x39x6mm
- 遥控范围：8-10m
- 电池：3V 纽扣式锂锰电池
- 红外载波频率：38KHz
- 面贴材料：0.125mm PET
- 有效寿命：20000 次以上

示例

- *第 14 课红外接收模块* (Mega 板项目)
- *第 14 课红外接收模块* （R3 板项目）

2.29 超声波模块

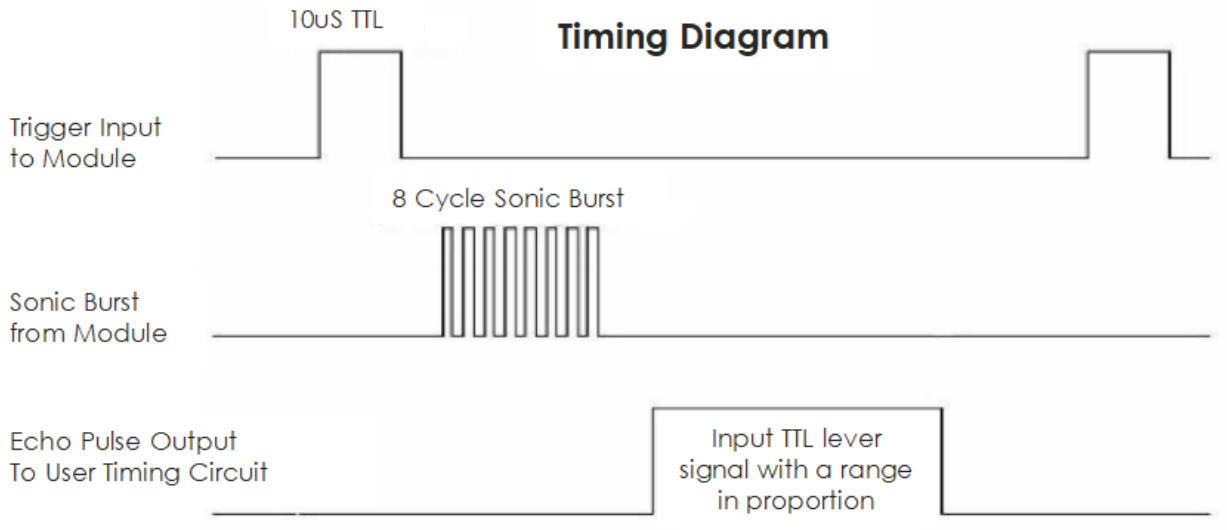
超声波传感器模块是一种使用超声波测量与物体之间距离的仪器。它有两个探头。一种是发送超声波，另一种是接收超声波并将发送和接收的时间转换为距离，从而检测设备与障碍物之间的距离。在实践中，它非常方便和实用。

提供 2cm-400cm 非接触测量功能，测距精度可达 3mm。可保证 5m 内信号稳定，5m 后信号逐渐减弱，直至 7m 位置消失。

该模块包括超声波发射器、接收器和控制电路。基本原则如下：

- 使用 IO 触发器处理至少 10us 的高电平信号。
- 模块自动发送 8 个 40khz 并检测是否有脉冲信号返回。
- 如果信号返回，通过高电平，则高输出 IO 持续时间就是超声波从发射到返回的时间。这里，测试距离 =（高时间 x 声速（340 m / s）/ 2。

时序图如下所示：



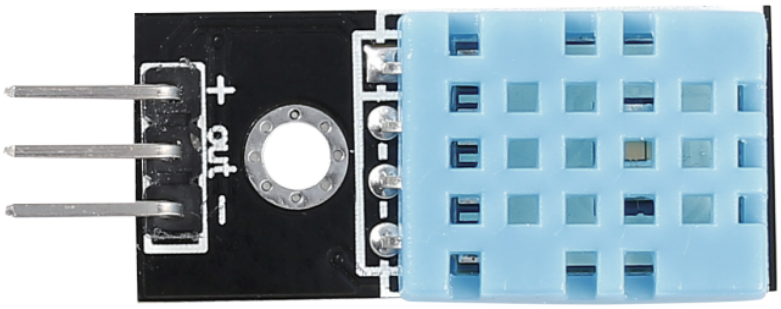
你只需要为触发输入提供一个 10us 的短脉冲即可开始测距，然后模块会以 40 kHz 的频率发出 8 个周期的超声波脉冲并提高其回波。你可以通过发送触发信号和接收回波信号之间的时间间隔来计算距离。

公式：us / 58 = 厘米或 us / 148 = 英寸；或：范围 = 高电平时间 * 速度（340M/S）/2；为防止触发信号与回波信号发生信号冲突，建议使用 60ms 以上的测量周期。

示例

- *第 13 课超声波* (Mega 板项目)
- *第 13 课超声波* （R3 板项目）
- *16. 游戏 - 愤怒的鹦鹉* (Scratch 项目)

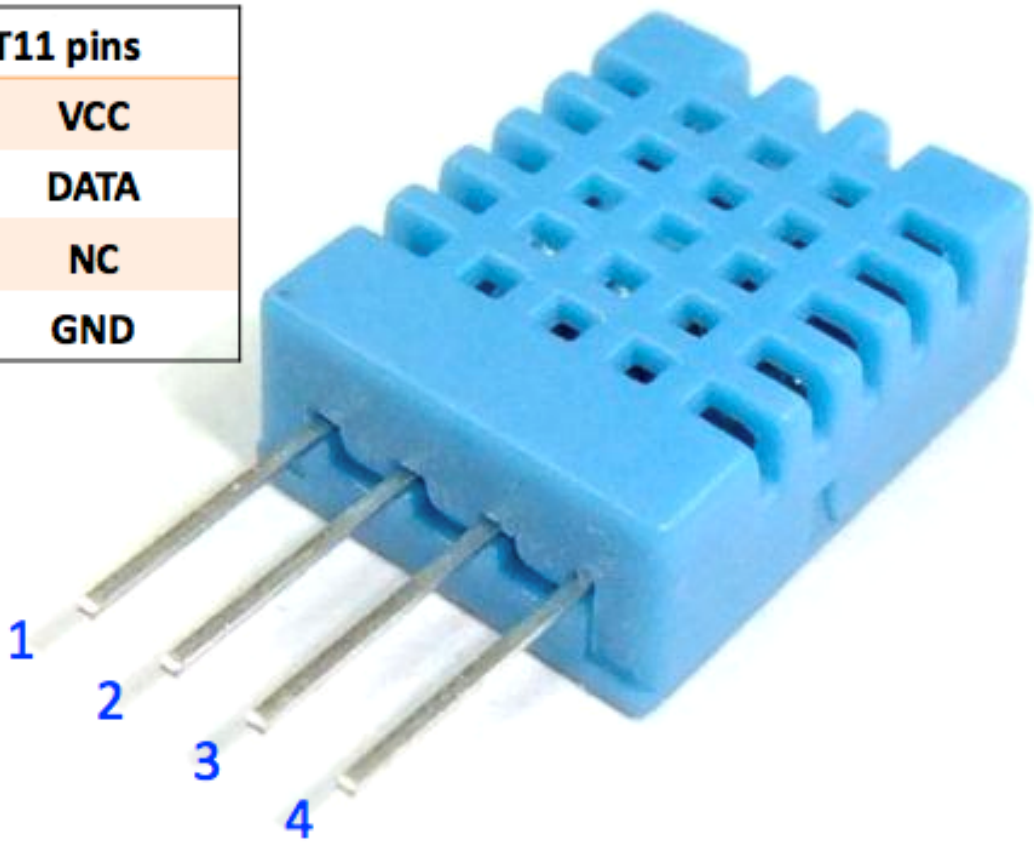
2.30 温湿度传感器模块



数字温湿度传感器 DHT11 是一种复合传感器，包含经过校准的温湿度数字信号输出。采用专用数字模块采集技术和温湿度传感技术，确保产品具有高可靠性和优异的长期稳定性。

只有三个引脚可供使用：VCC、GND 和 DATA。通信过程开始于 DATA 线向 DHT11 发送启动信号，DHT11 接收信号并返回应答信号。然后主机收到应答信号，开始接收 40 位温湿度数据（8 位湿度整数 +8 位湿度小数 +8 位温度整数 +8 位温度小数 +8 位校验和）。

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



- DHT11 数据表

示例

- 第 *15* 课温湿度传感器 (Mega 板项目)
- 第 *15* 课温湿度传感器 （R3 板项目）
- *11.* 读取温湿度 （Scratch 项目）

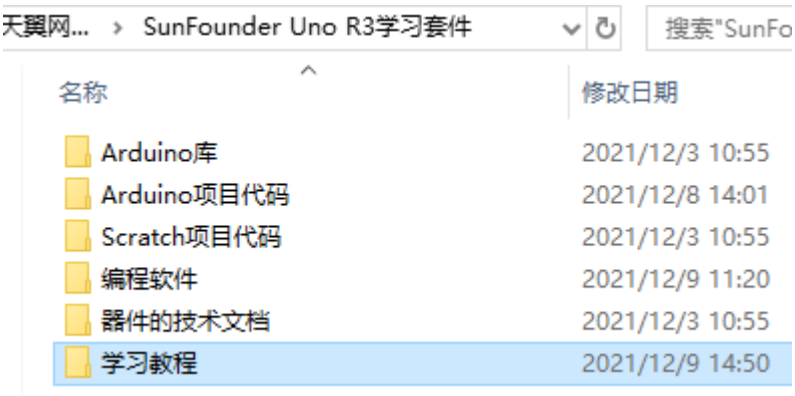
CHAPTER 3

下载资料

我们已经将所有相关的资料都已经上传到天翼云盘，下面是链接和访问码，进去之后根据需要选择相应的压缩包进行下载。

- `SunFounder Uno R3 学习套件资料下载`
- （访问码：nn2y）

下载并解压后，你会看到以下文件夹：



- **Arduino 库**：包含了 4 个需要单独添加到 Arduino IDE 的库，在章节*添加库* 中会用到。
- **Arduino 项目代码**：包含了 23 个 Arduino 项目的代码文件，适用于*Arduino 项目 - Uno R3* 和*Arduino 项目 - Mega2560* 章节。
- **Scratch 项目代码**：包含了 19 个 Scratch 项目的代码和图片，适用于*Scratch 项目* 章节。
- **编程软件**：包含了 **Arduino IDE** 和 **PictoBlox** 在不同系统的安装包。
- **器件的技术文档**：在你想要深入了解某个器件时，你可以到这个文件夹学习器件的专业技术资料。
- **参考电子书**：提供的是其他相关的电子书，仅供参考。
- **xxx 套件说明书**：这是整个套件 PDF 版本的说明书，无法保证是最新版，建议以在线教程以准。

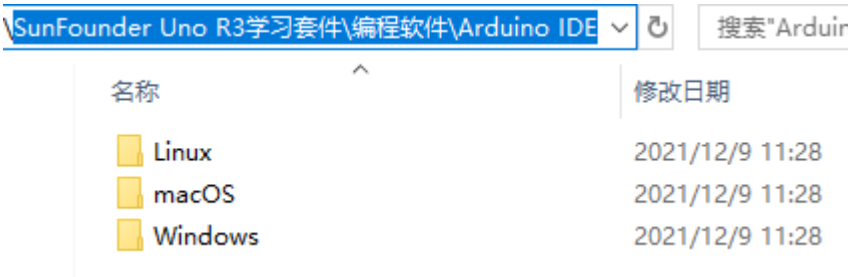
CHAPTER 4

安装和介绍 Arduino IDE

Arduino 是一个软件和硬件简单的开源平台。即使您是初学者，也可以在短时间内掌握它。提供集成开发环境（IDE）进行代码编译，兼容多种控制板。所以你只需下载 Arduino IDE，将代码文件上传到 Arduino 板上，然后你就可以看到相关的实验现象。有关更多信息，请参阅 https://www.arduino.cc/。

4.1 下载 Arduino IDE

参考章节下载资料 来下载相关的资料，然后进入到 `SunFounder Uno R3 学习套件\编程软件\Arduino IDE` 路径中。在这个文件夹中，包含了 Arduino IDE 在不同系统的安装包，请根据你的系统来选择。



当然，如果你的英语还不错，你可以去 Arduino 官网：https://www.arduino.cc/en/software，下载最新版本的包装包。

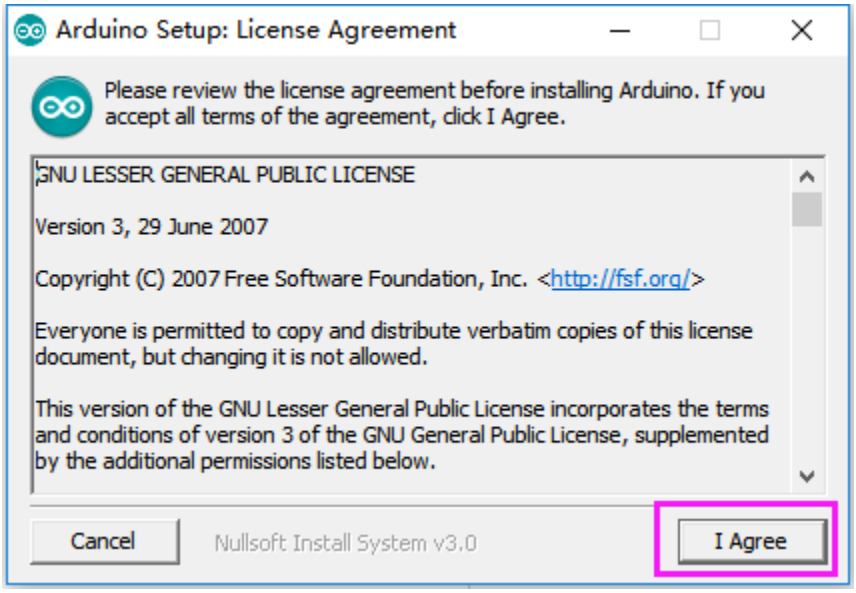
4.2 Windows 安装步骤

下面是 Windows 系统的安装步骤，对于其他系统的安装教程，请参考：

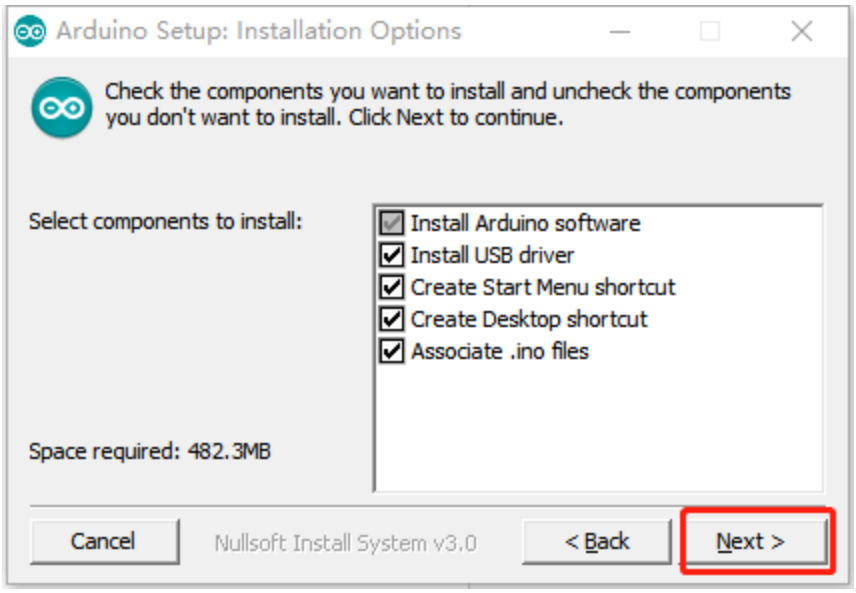
- macOS
- Linux

第 1 步：进入到 `SunFounder Uno R3 学习套件\编程软件\Arduino IDE\Windows` 路径中，双击 `arduino-x.x.xx-windows.exe` 文件。

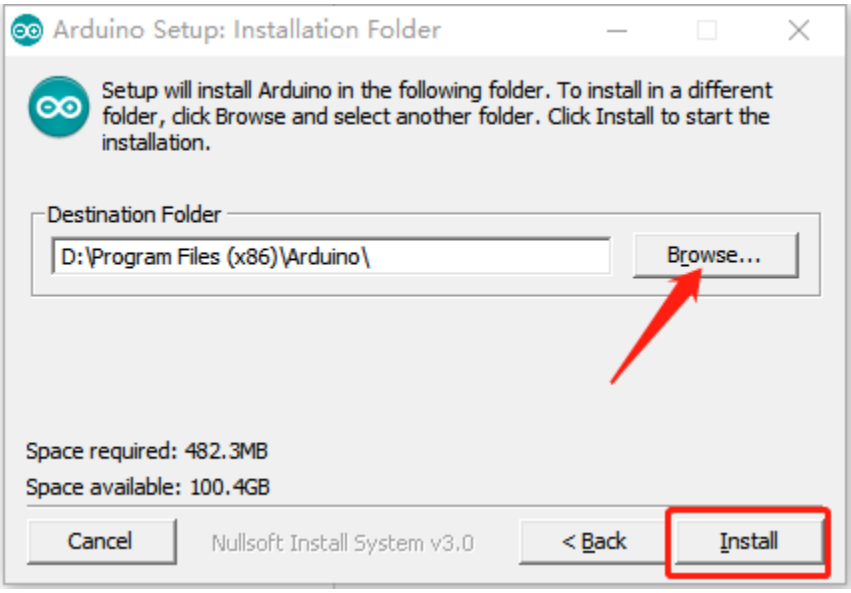
第 2 步：在弹出的窗口，点击 **I Agree**。



第 3 步：点击 `Next`。



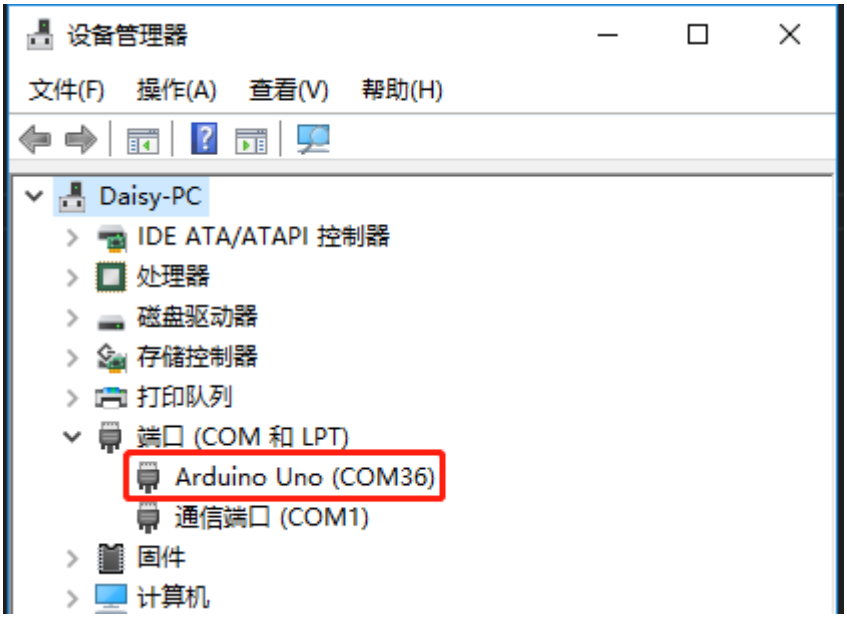
第 4 步：选择安装路径，默认情况下是安装在 C 盘。你可以点击 `Browse` 来选择其他的安装路径，然后点击 `OK`。



第 5 步：在安装过程中，会有窗口弹出安装需要的驱动，请选择‘Always trust software from “Arduino LLC”’。安装完成后，点击 **Close**。

第 6 步：用一根 USB 线将 Arduino 板插入到电脑的 USB 端口，在左下角搜索设备管理器，双击来将它打开。

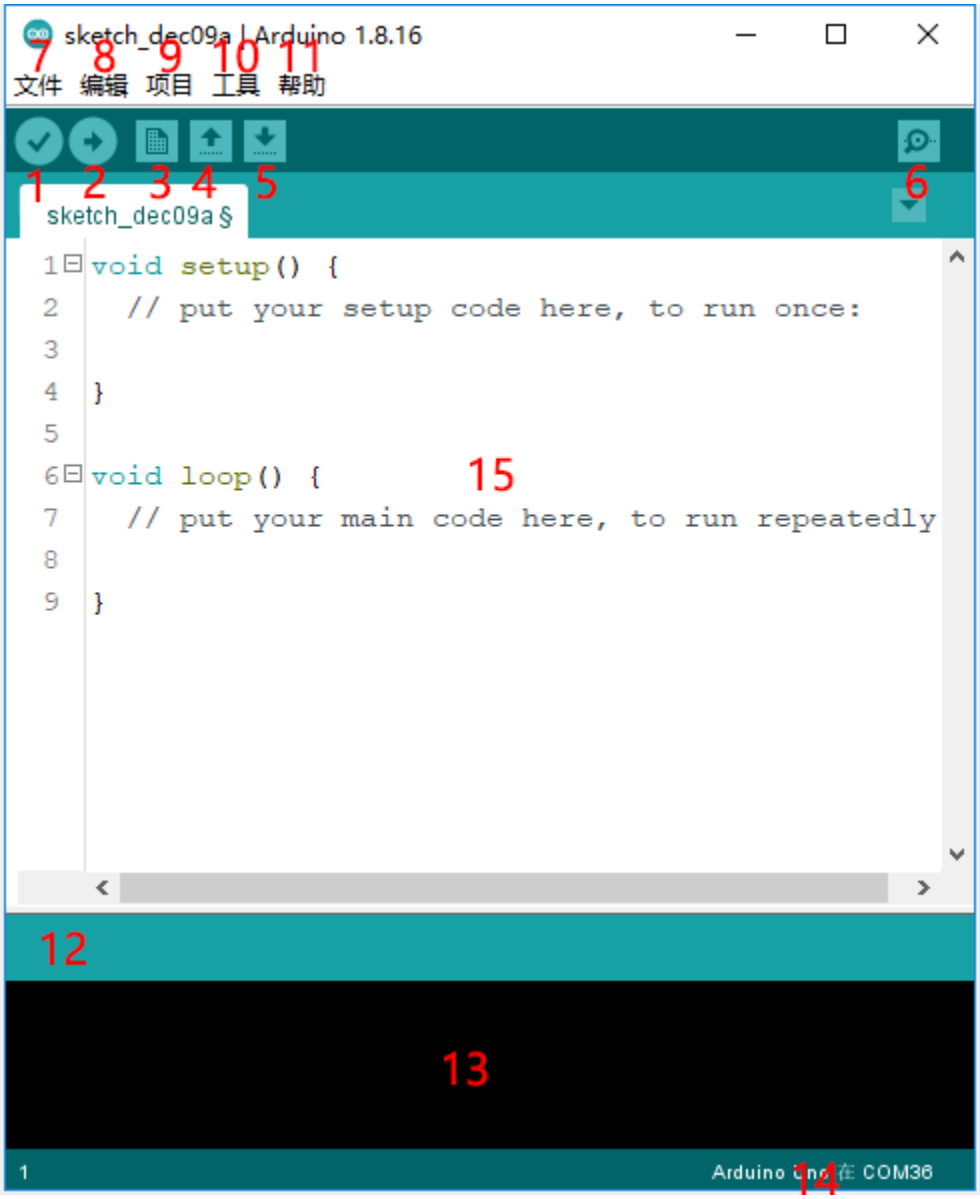
第 7 步：如果在端口处看到 Arduino Uno (COMxx) 或者 Arduino Mega2560 (COMxx)，说明电脑已经正确识别的你的 Arduino 板，请记住这个端口号。



4.3 介绍 Arduino IDE

双击安装过程创建的 Arduino 图标（arduino.exe）。

然后将出现 Arduino IDE。现在让我们来看下这个编程软件的基本介绍。



- 1、验证：编译您的代码，任何语法问题都会提示错误。
- 2、上传：将代码上传到您的开发板。当您单击按钮时，板上的 RX 和 TX LED 将快速闪烁，并且在上传完成之前不会停止。
- 3、新建：创建一个新的代码编辑窗口。
- 4、打开：打开 Arduino 代码文件。
- 5、保存：保存代码。
- 6、串行监视器：单击按钮，将出现一个窗口。它接收从您的控制板发送的数据。这对于调试非常有用。
- 7、文件：点击菜单，会出现一个下拉列表，包括文件的创建、打开、保存、关闭、一些参数配置等。
- 8、编辑：单击菜单。在下拉列表中，有剪切、复制、粘贴、查找等一些编辑操作，以及相应的快捷键。
- 9、项目: 包括验证、上传、添加文件等操作。更重要的功能是包含库——您可以在其中添加库。
- 10、工具：包括一些工具——最常用的 Board（你使用的板）和 Port（你的板所在的端口）。每次要上

传代码时，都需要选择或检查它们。

- 11、帮助：如果您是初学者，您可以查看菜单下的选项并获得您需要的帮助，包括在 IDE 中的操作、介绍信息、故障排除、代码解释等。
- 12、在此消息区，无论何时编译或上传，都会出现摘要消息。
- 13、编译和上传过程中的详细信息。比如使用的文件在哪个路径，错误提示的详细信息。
- 14、开发板和端口: 这里可以预览选择的开发板和端口。如果有任何不正确，您可以通过工具 -> 开发板或端口再次选择它们。
- 15、IDE 的编辑区：你可以在这里写代码。

CHAPTER 5

添加库

5.1 什么是库？

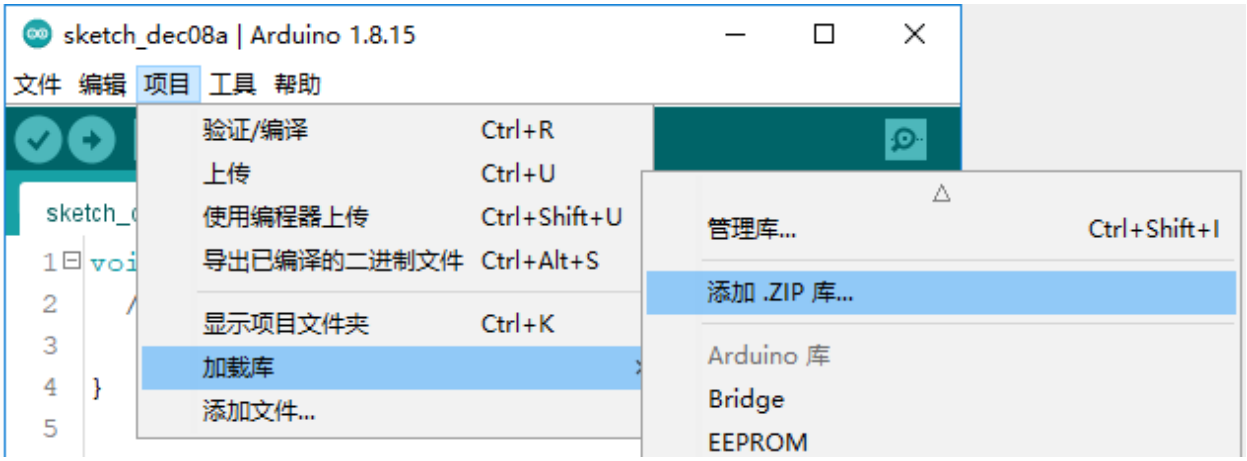
一个库，收集一些函数定义和头文件，通常包含两个文件：.h（头文件，包括函数语句、宏定义、构造函数定义等）和.cpp（执行文件，包括函数实现、变量定义和很快）。当需要使用某个库中的函数时，只需要添加一个头文件（例如 `#include <dht.h>`），然后调用该函数即可。这可以使您的代码更加简洁。如果不想使用该库，也可以直接编写该函数定义。尽管结果是代码会很长并且不方便阅读。

添加库 Arduino IDE 中已经内置了一些库，但可能需要添加其他一些库。那么现在让我们看看如何添加一个。有 3 种方法

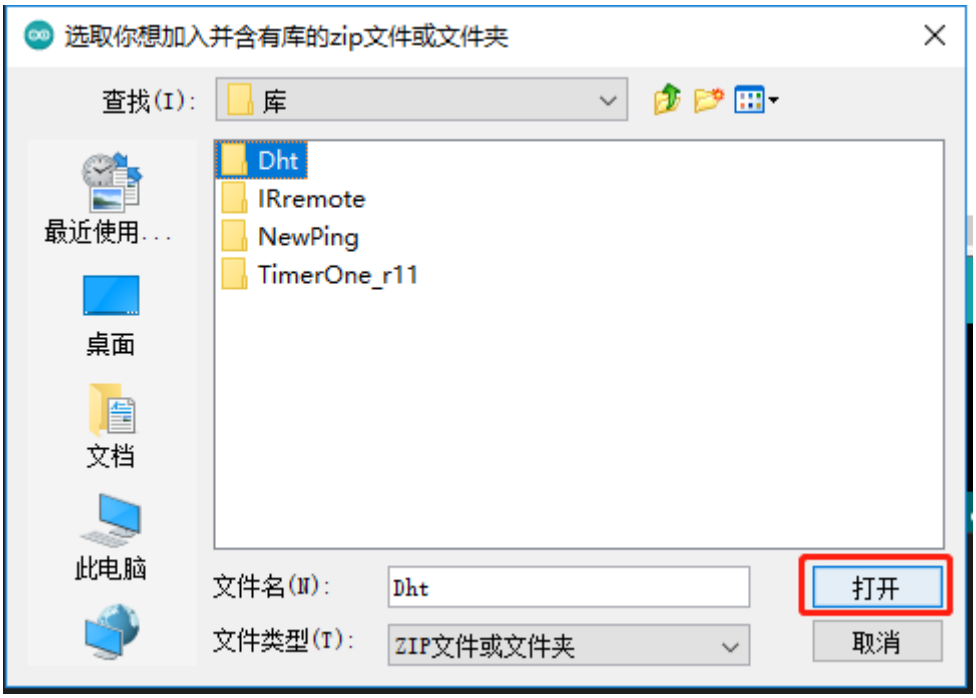
5.1.1 方法一

直接在 Arduino IDE 中导入库（以下以 Dht 为例）。这种方法的优点是易于理解和操作，但另一方面，一次只能导入一个库。所以当你需要添加相当多的库时是不方便的。

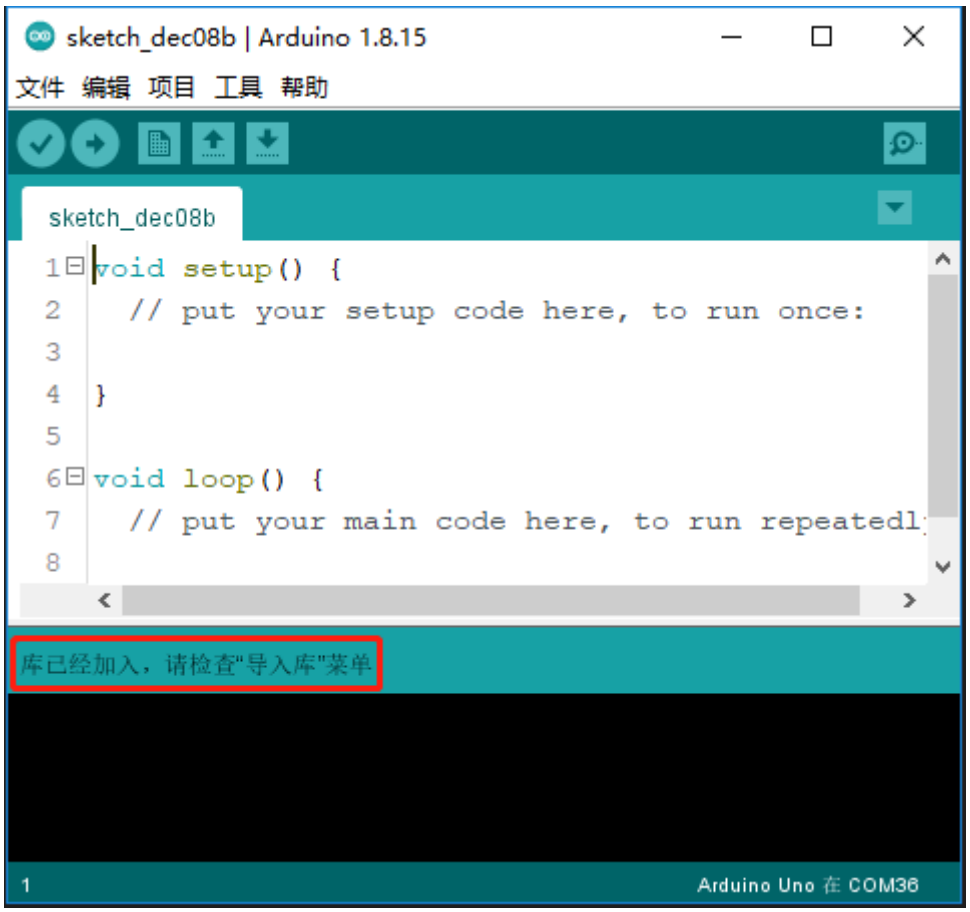
第 1 步：选择 **项目 -> 加载库 -> 添加.ZIP 库**。



第 2 步：进入到 `SunFounder Uno R3 学习套件\Arduino 库路径`, 请确保你已经参考了下载资料 下载了相关资料。



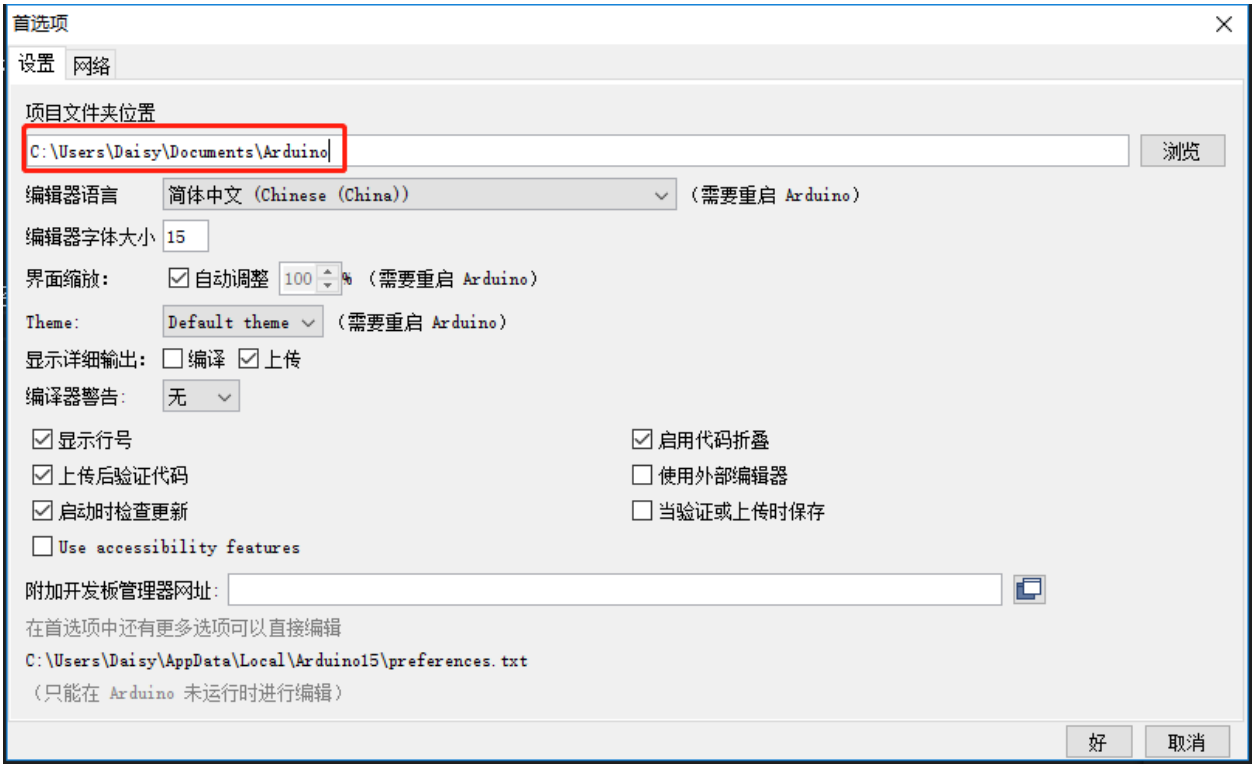
第 3 步：当您看到“库已经加入，请检查“导入库”菜单”，表示您已成功添加库。然后请使用相同的方法添加其他库。



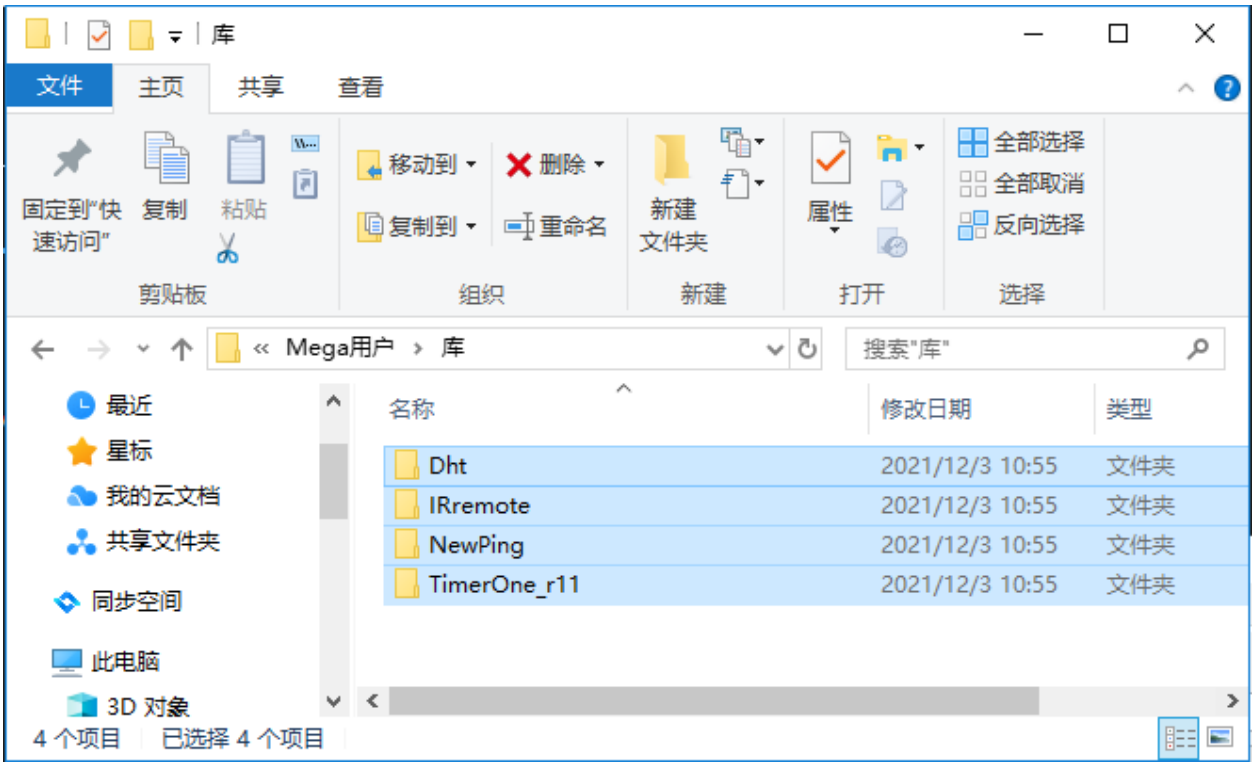
5.1.2 方法二

直接将库复制到 `Arduino/libraries` 路径。这种方法可以复制所有的库，一次添加，但缺点是这个路径比较难找到。

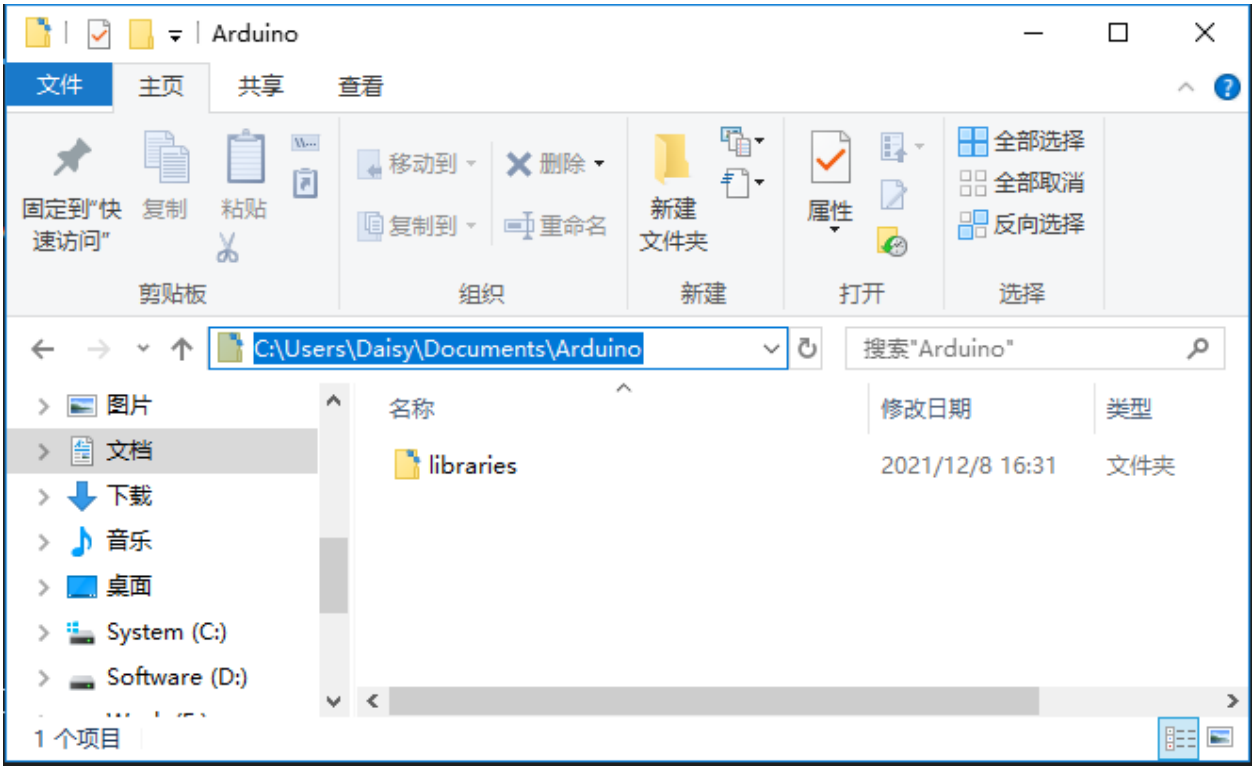
第 1 步：点击 **文件 -> 首选项**，在弹出的窗口中，找到项目文件夹位置。



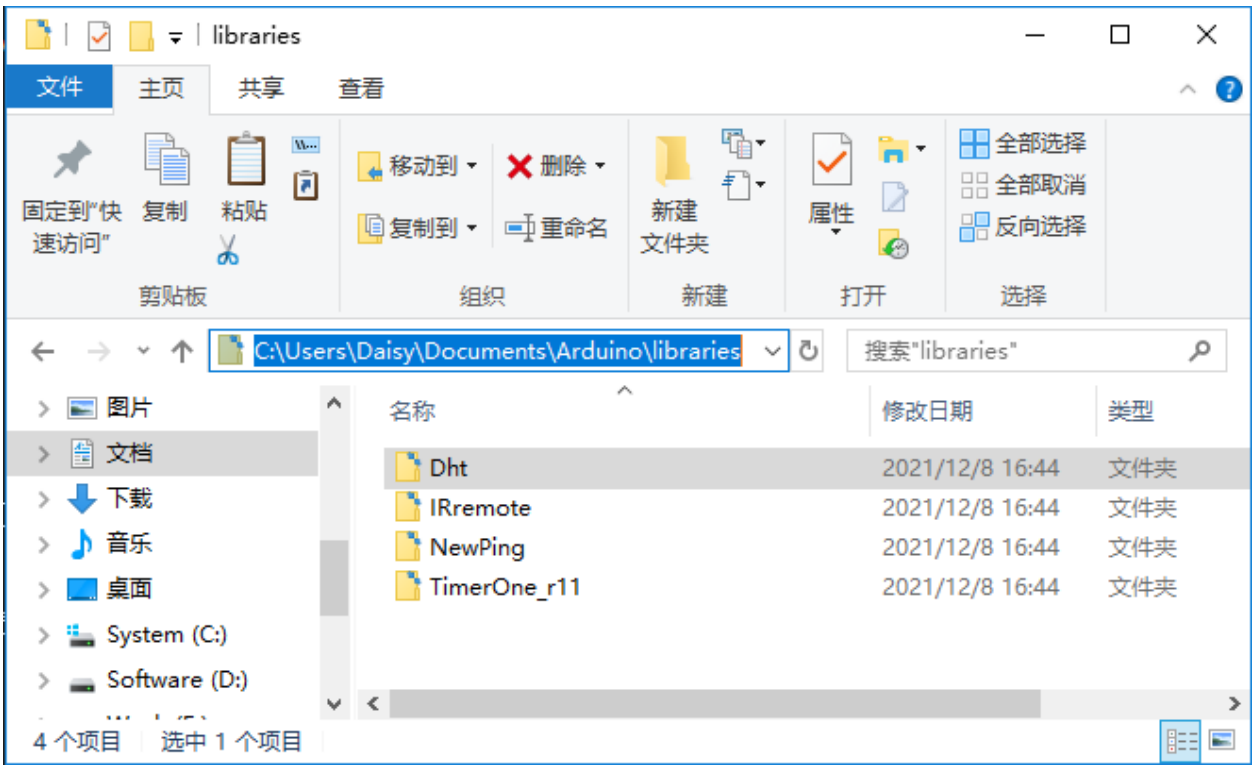
第 2 步：进入 `SunFounder Uno R3 学习套件\Arduino 库路径`，复制所有的文件夹。



第 3 步：进入第一步所示的路径，你会看到有一个 `libraries` 文件夹，将它打开。



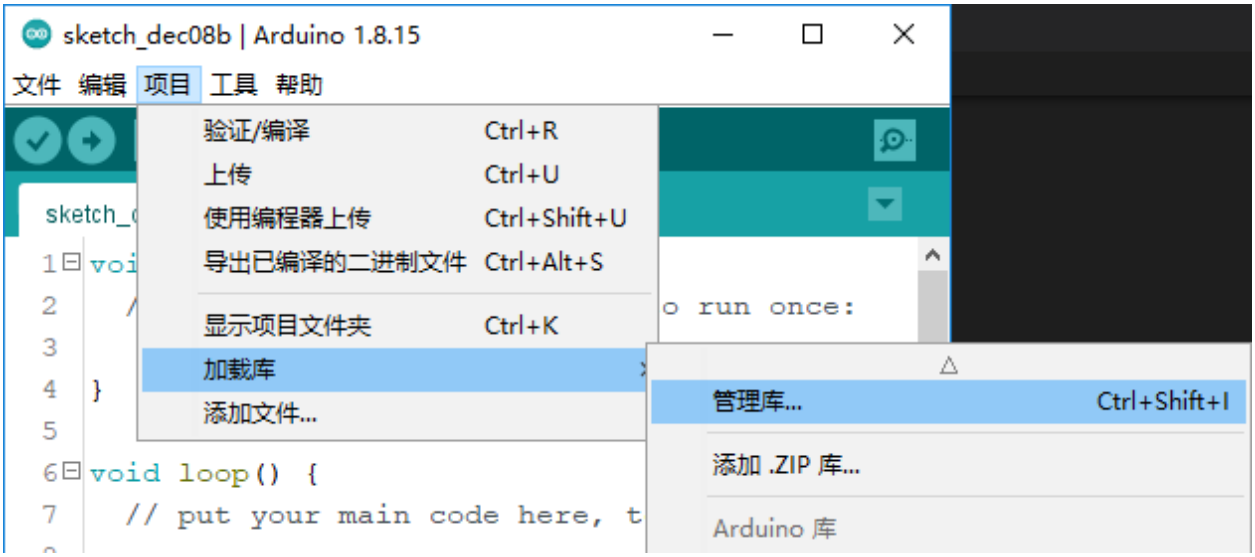
第 4 步：将之前复制的所有库粘贴到该文件夹 中。然后你可以在库文件夹中看到它们。



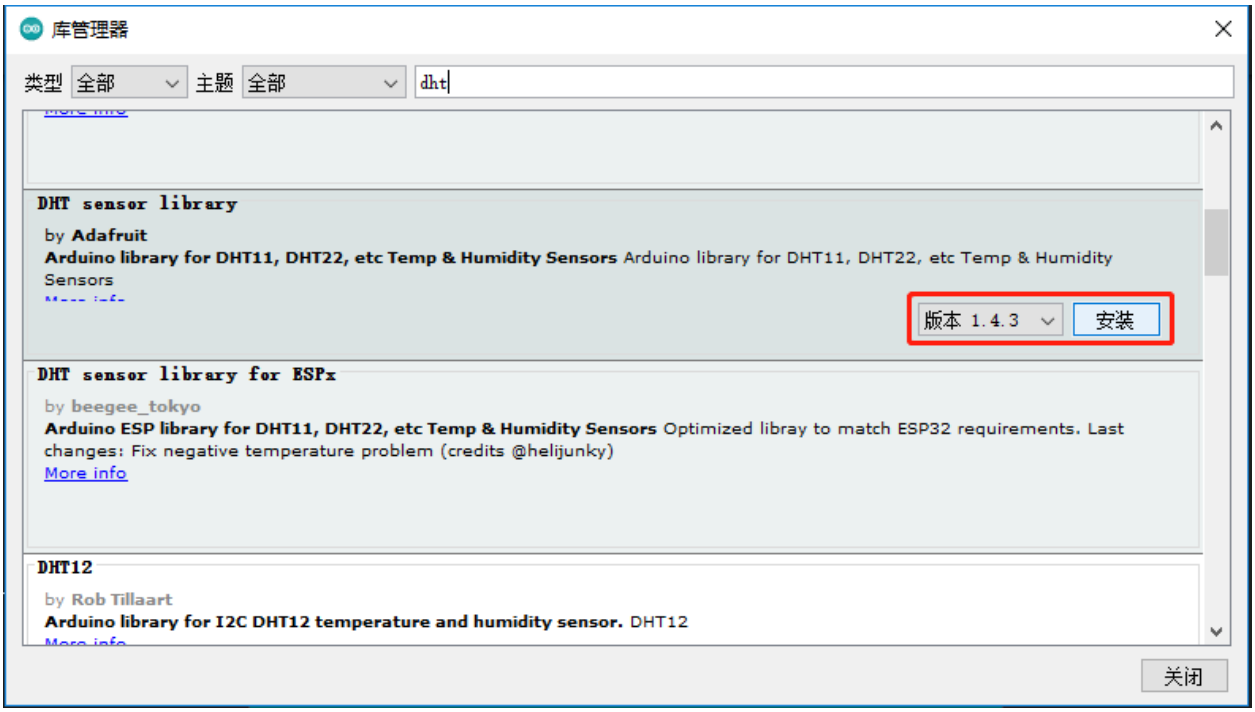
5.1.3 方法三

使用库管理器将新库安装到 Arduino IDE 中。不是很推荐用这个方法，因为使用这个方法添加的库可能会和我们提供的示例代码不兼容。

第 1 步：点击 **项目 -> 加载库-> 管理库**。



第 2 步：输入库名搜索，如 dht，可以选择最新版本安装。



完成后，您可以关闭库管理器。

CHAPTER 6

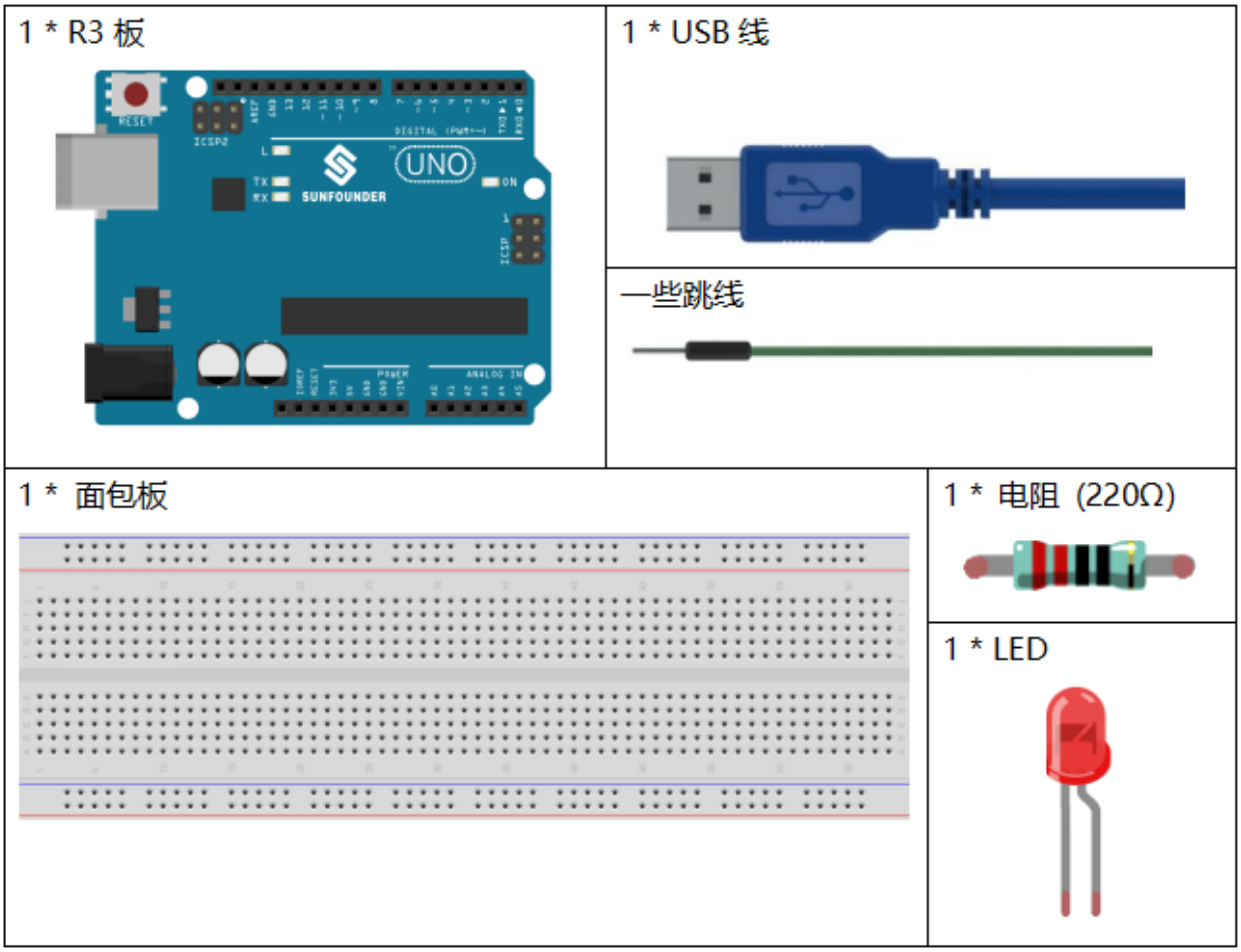
Arduino 项目 - Uno R3

6.1 第 1 课闪烁的 LED

6.1.1 介绍

你之前应该已经学会了如何安装 Arduino IDE 并添加库。现在你可以从一个简单的实验开始，学习 IDE 中的基本操作和代码。

6.1.2 所需器件

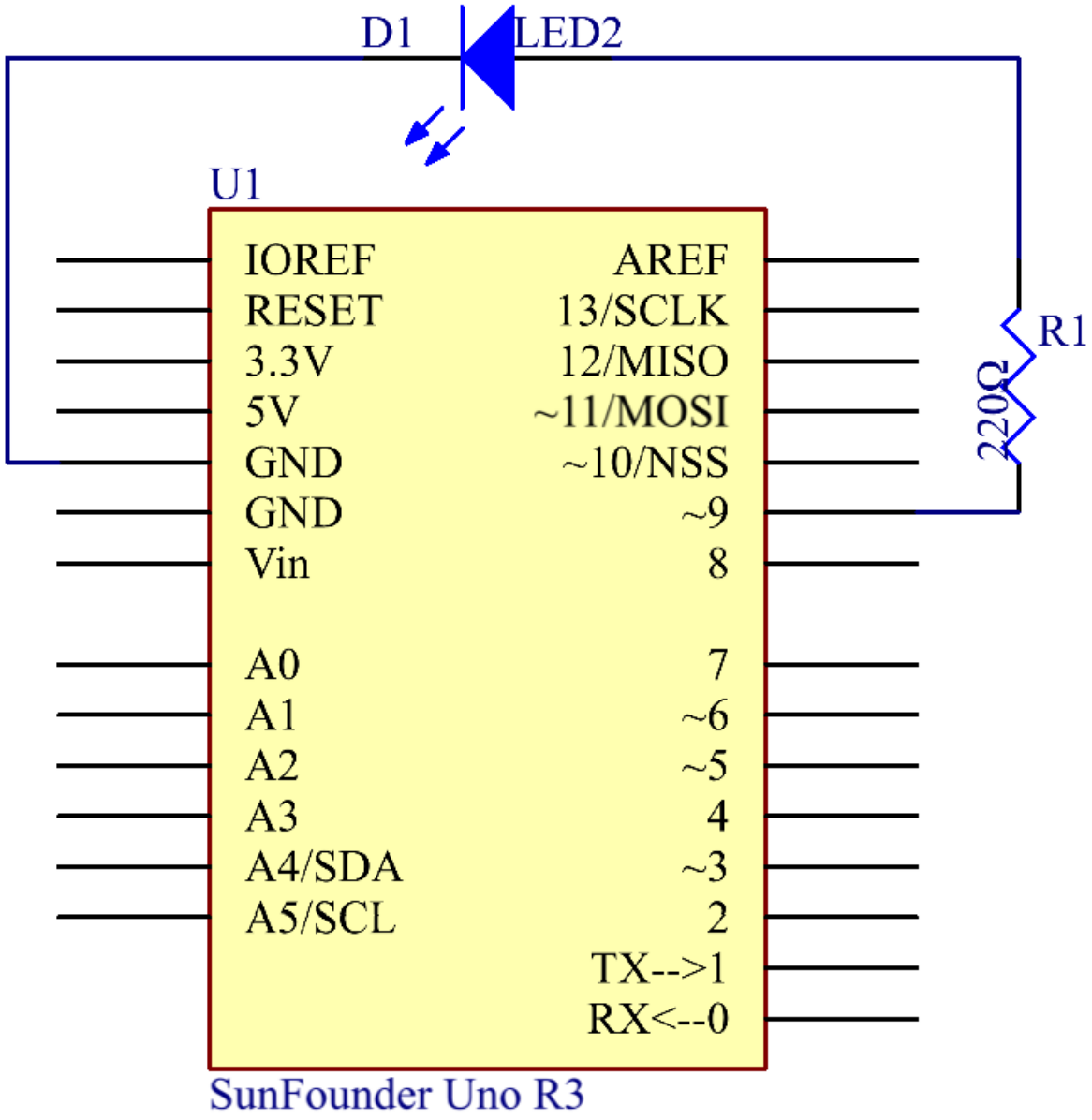


- *SunFounder R3 板*
- *面包板*
- *跳线*
- *LED 发光二极管*
- *电阻*

6.1.3 原理图

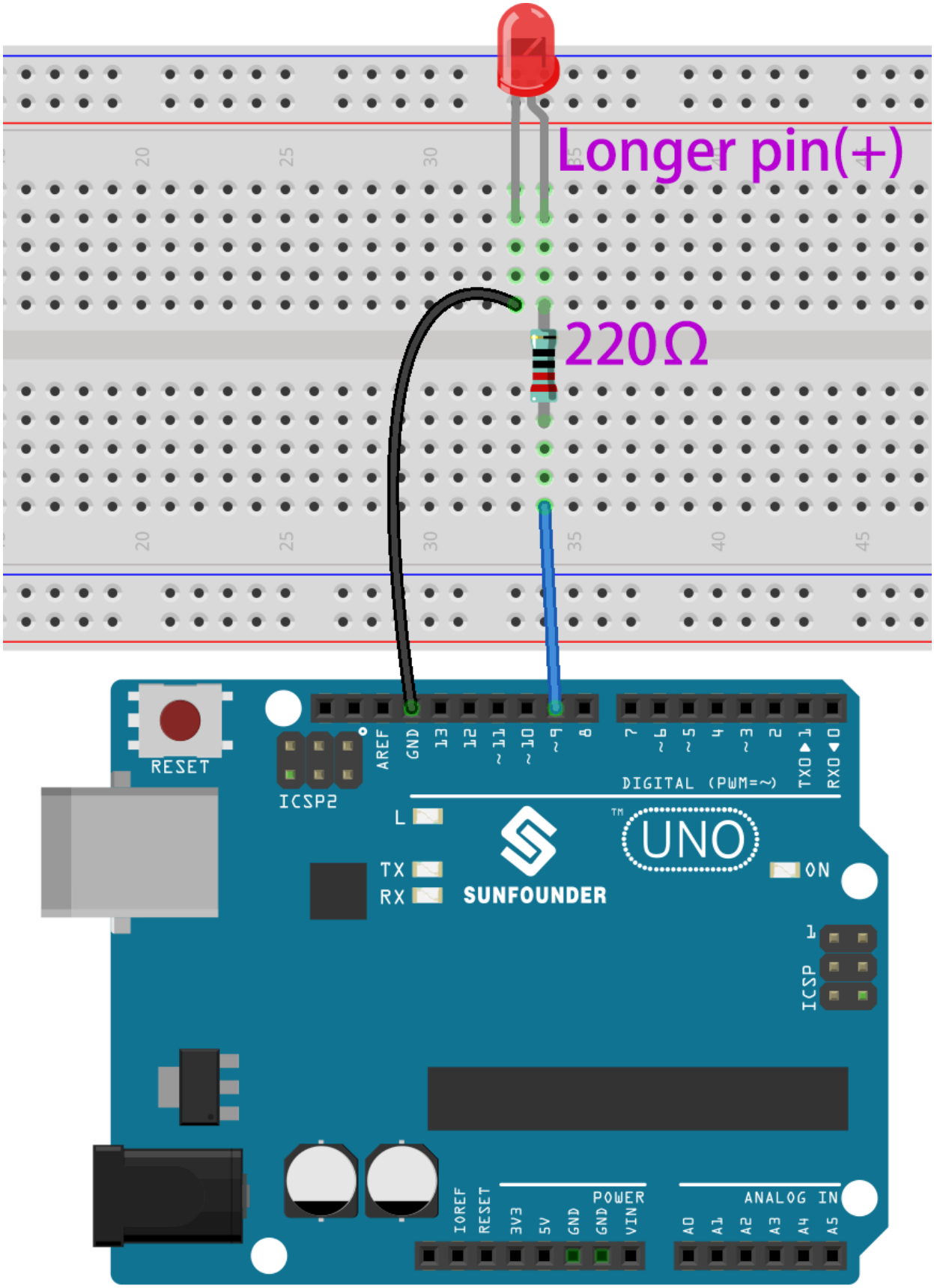
将 220ohm 电阻的一端接板子的 9 脚，另一端接 LED 的正极（长脚），LED 的负极（短脚）接 GND。当 9 脚输出高电平时，电流通过限流电阻到达 LED 的正极。由于 LED 的阴极连接到 GND，因此 LED 会亮起。当引脚 9 输出低电平时，LED 熄灭。

原理图如下所示：



6.1.4 实验步骤

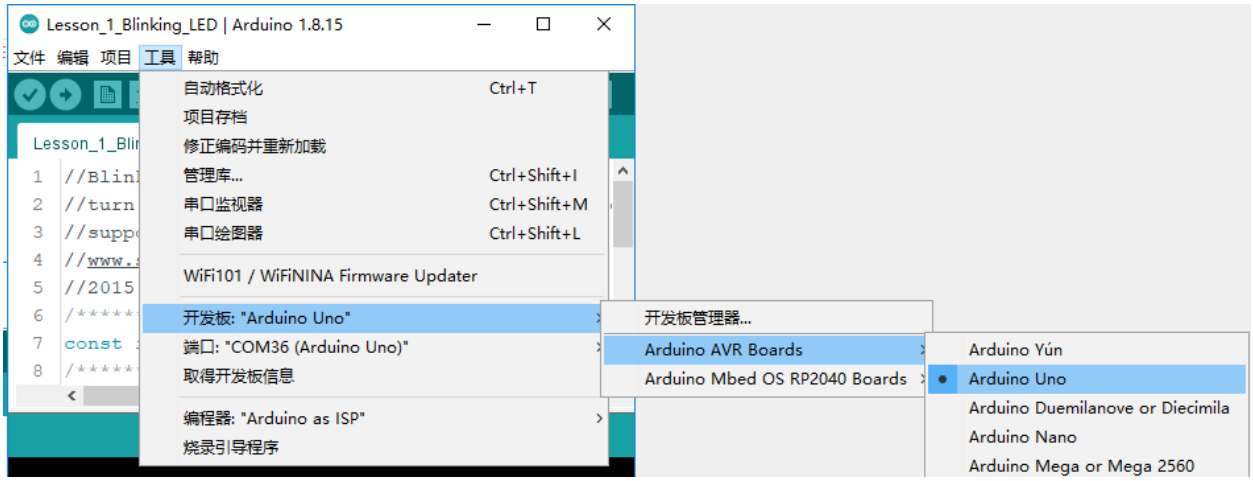
第 1 步：搭建电路。(LED 上长的引脚为阳极)。然后用一根 USB 线将板子插入到电脑上。


Longer pin(+)
220Ω
RESET
ICSP2
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
1
0
TXD
RXD
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

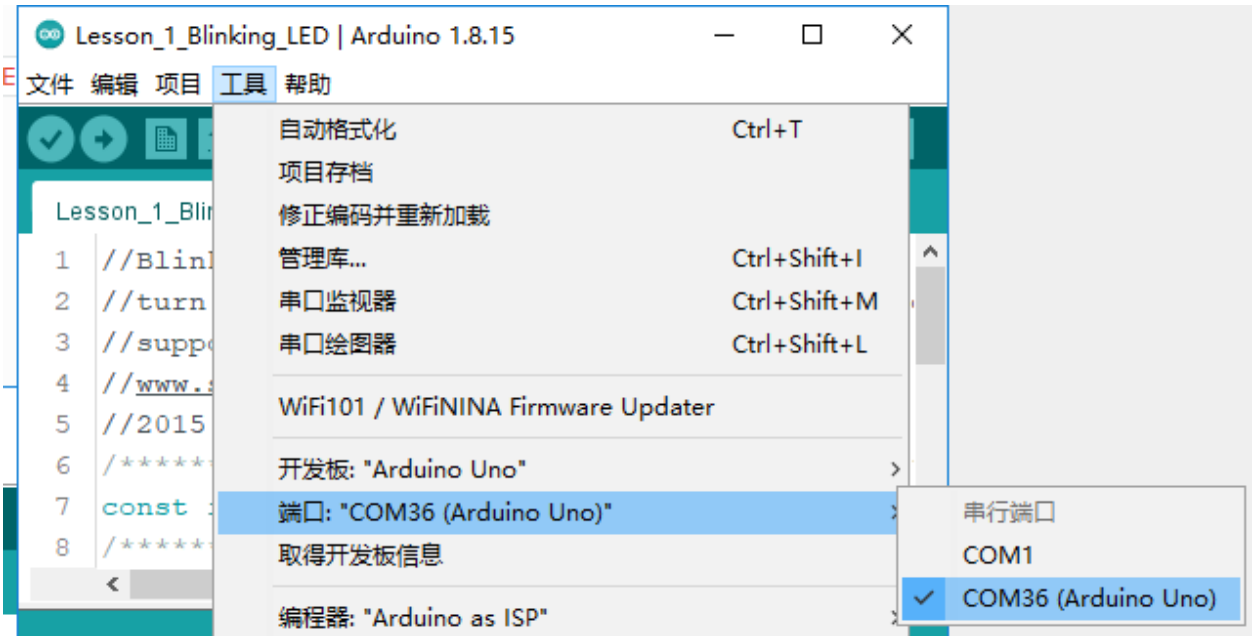
第 2 步：打开路径 `SunFounder Uno R3 学习套件\Arduino 项目代码\Lesson_1_Blinking_LED` 中的代码文件 `Lesson_1_Blinking_LED.ino`。

第 3 步：在上传代码前，你需要选择正确的开发板和端口。

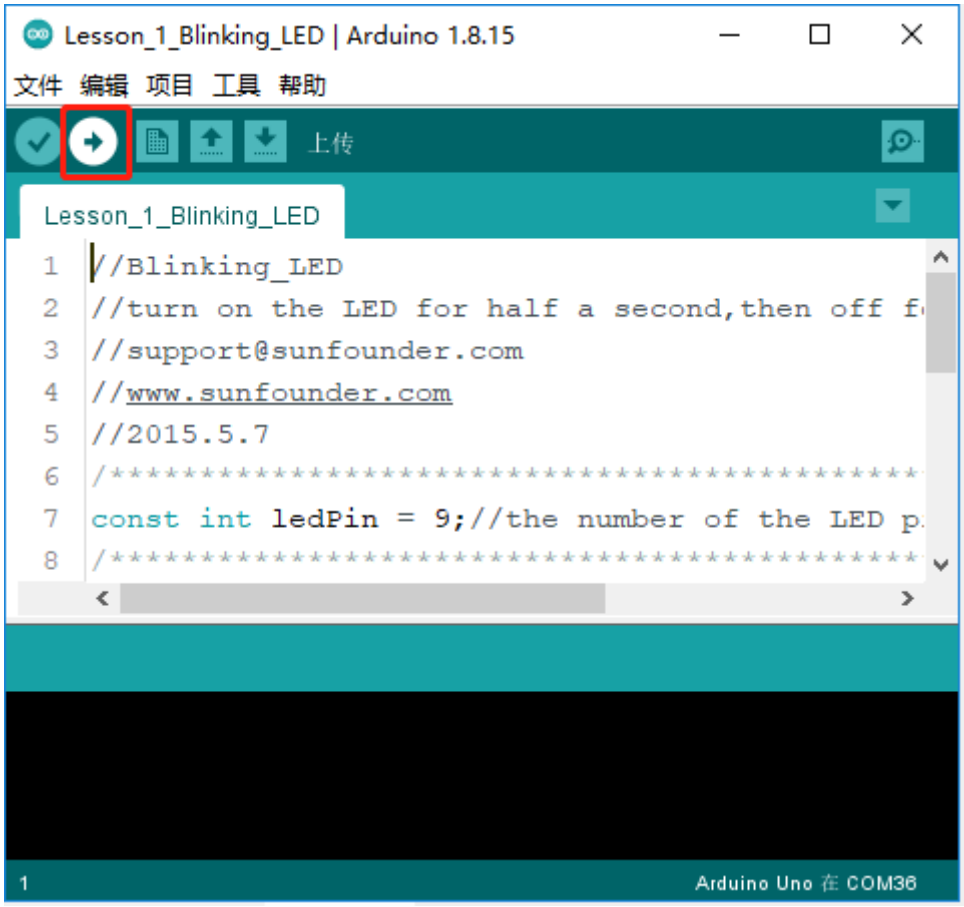
点击 **工具 -》开发板**，然后选择 **Arduino Uno**。



然后再点击 **工具 -》端口**，你的端口应该和我的不一样。

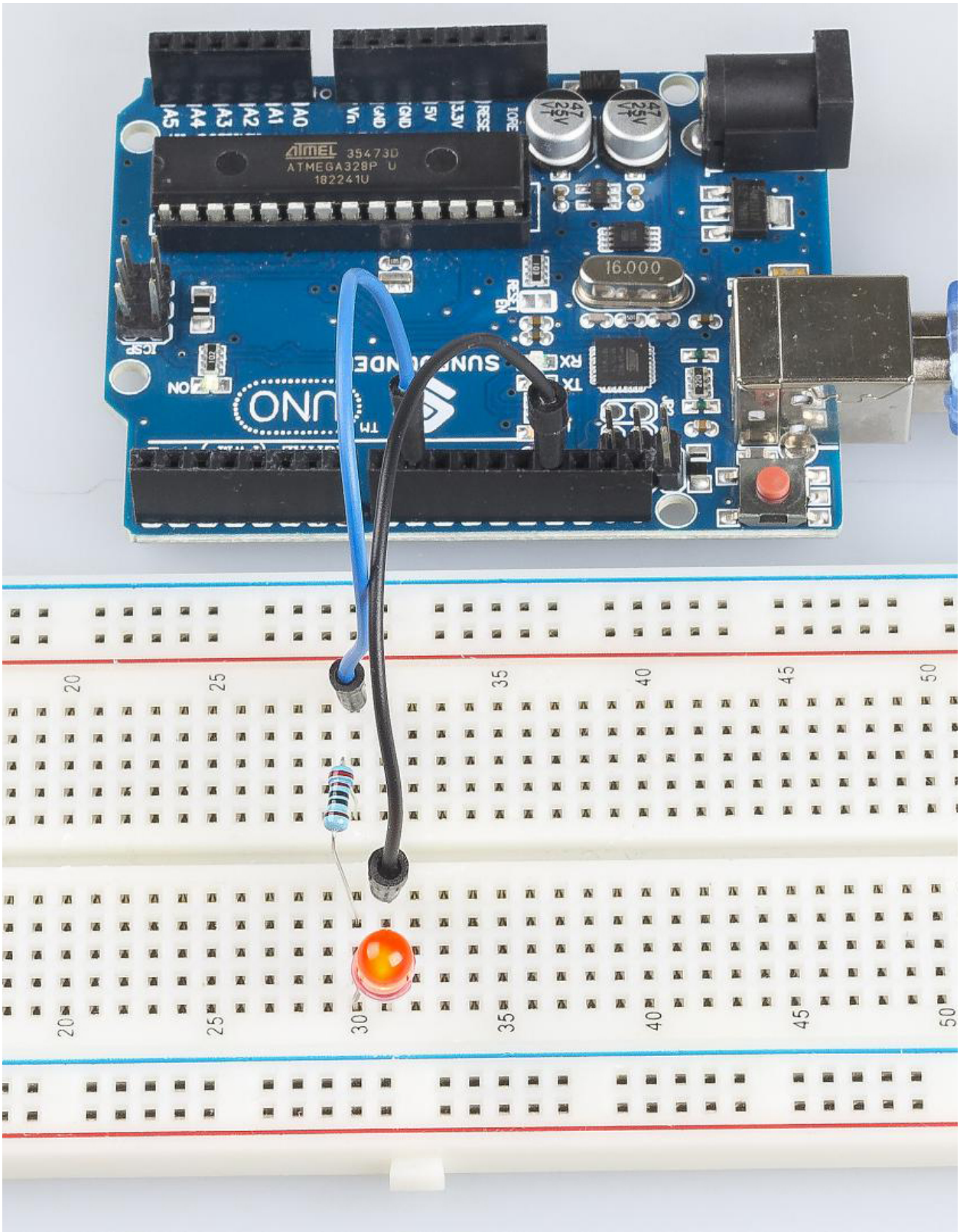


第 4 步：点击 **上传**按钮来将代码上传到主板上。



第 5 步：如果 **上传成功**提示出现，代表代码已成功上传到主板上。

现在你将看到 LED 闪烁。


ATMEL
ATMEGA328P
16.000
UNO

6.1.5 代码

6.1.6 代码分析

定义变量

```
const int ledPin = 9; //the number of the LED pin
```

你应该在使用前定义每个变量，以防出错。该行为引脚 9 定义了一个常量变量 ledPin。在下面的代码中，ledPin 代表引脚 9。你也可以直接使用引脚 9 代替。

setup() 函数

一个典型的 Arduino 程序由两个子程序组成：用于初始化的 `setup()` 和包含程序主体的 `loop()`。

- `setup()`：该函数通常用于初始化数字引脚，并将它们设置为输入或输出，以及串行通信的波特率。
- `loop()`：该函数包含了整个代码运行顺序，将循环运行，除非发生停电之类的事情，否则它不会停止。

```
void setup()
{
    pinMode(ledPin,OUTPUT);//initialize the digital pin as an output
}
```

在 `setup()` 函数中将 `ledPin` 设置为输出。

- `pinMode(Pin)`：将指定的引脚配置为输入或输出。

setup 之前的 void 意味着这个函数不会返回值。即使不需要初始化引脚，你仍然需要此功能。否则编译会出错。

loop() 函数

```
void loop()
{
    digitalWrite(ledPin,HIGH);//turn the LED on
    delay(500);               //wait for half a second
    digitalWrite(ledPin,LOW); //turn the LED off
    delay(500);               //wait for half a second
}
```

本程序是设置 `ledPin` 为 `HIGH` 来让 LED 点亮，使用 `delay()` 函数来设置点亮时间，单位为毫秒。同样，设置为 `LOW` 将让 LED 熄灭，时间为 500 毫秒。代码上传之后，你将看到 LED 点亮 500 毫秒 (0.5 秒)，熄灭 500 毫秒（0.5 秒），这种交替不会停止，除非断电。

- `digitWrite()`：写一个 `HIGH` 或 `LOW` 值到数字引脚。当此引脚在 `pinModel()` 函数中设置为输出时，其电压将设置为相应的值：5V（或 3.3V 板上的 3.3V）代表高，0V（地）代表低。

6.1.7 实验总结

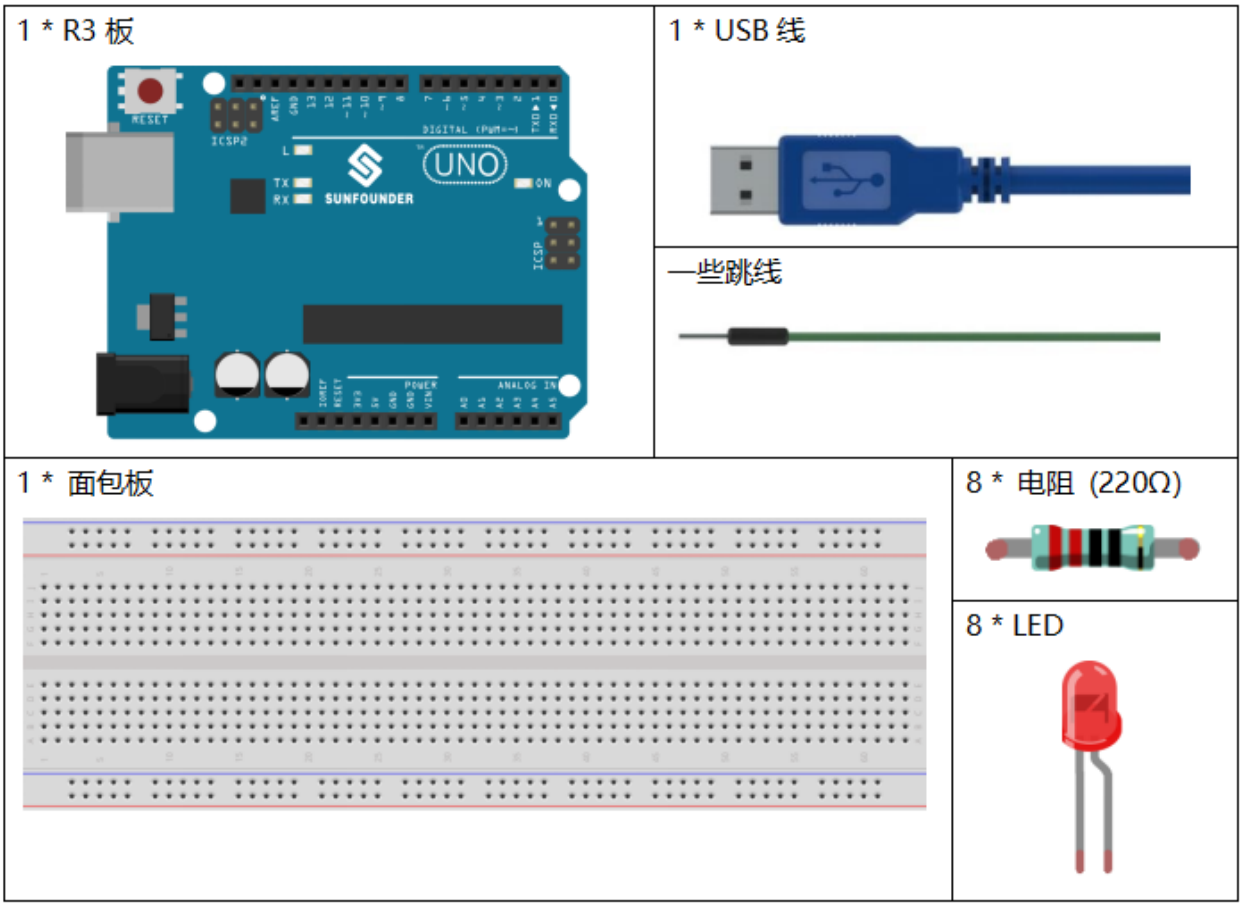
通过这个实验，你已经学会了如何打开 LED。你还可以通过更改 `delay (num)` 中的 num 值来更改 LED 的闪烁频率。例如，将其更改为 `delay(250)`，你会发现 LED 闪烁更快。

6.2 第 2 课流水灯

6.2.1 介绍

在本课中，我们将进行一个简单而有趣的实验——使用 LED 创造流动的 LED 灯。顾名思义，这 8 颗排成一排的 LED 依次亮起、变暗，就像流水一样。

6.2.2 所需器件

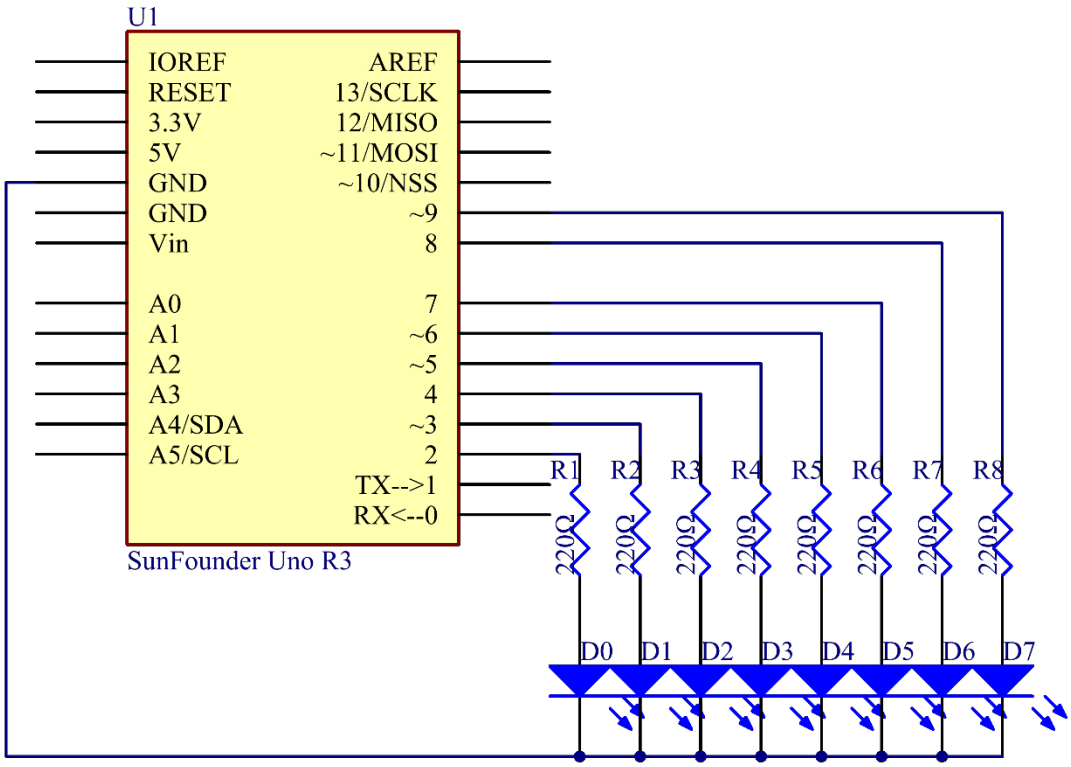


- *SunFounder R3 板*
- 面包板
- 跳线
- *LED 发光二极管*
- 电阻

6.2.3 原理图

这个实验的原理很简单，就是依次点亮 8 个 LED。8 个 LED 分别连接到引脚 2~9。将它们设置为高电平，引脚上相应的 LED 将亮起。控制每个 LED 亮起的时间，你将看到流动的 LED 灯。

原理图如下所示：



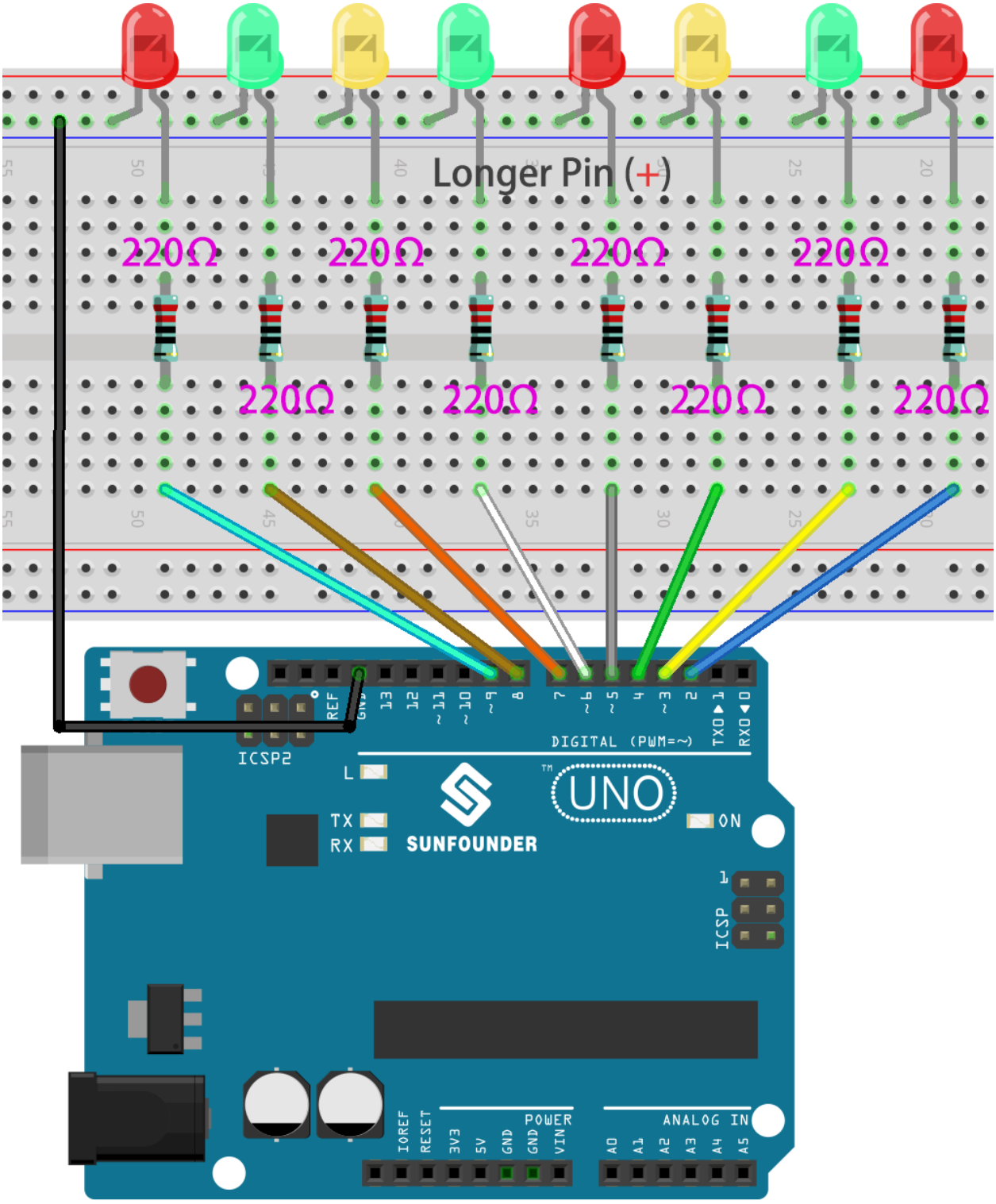
6.2.4 实验步骤

第 1 步： 搭建电路.

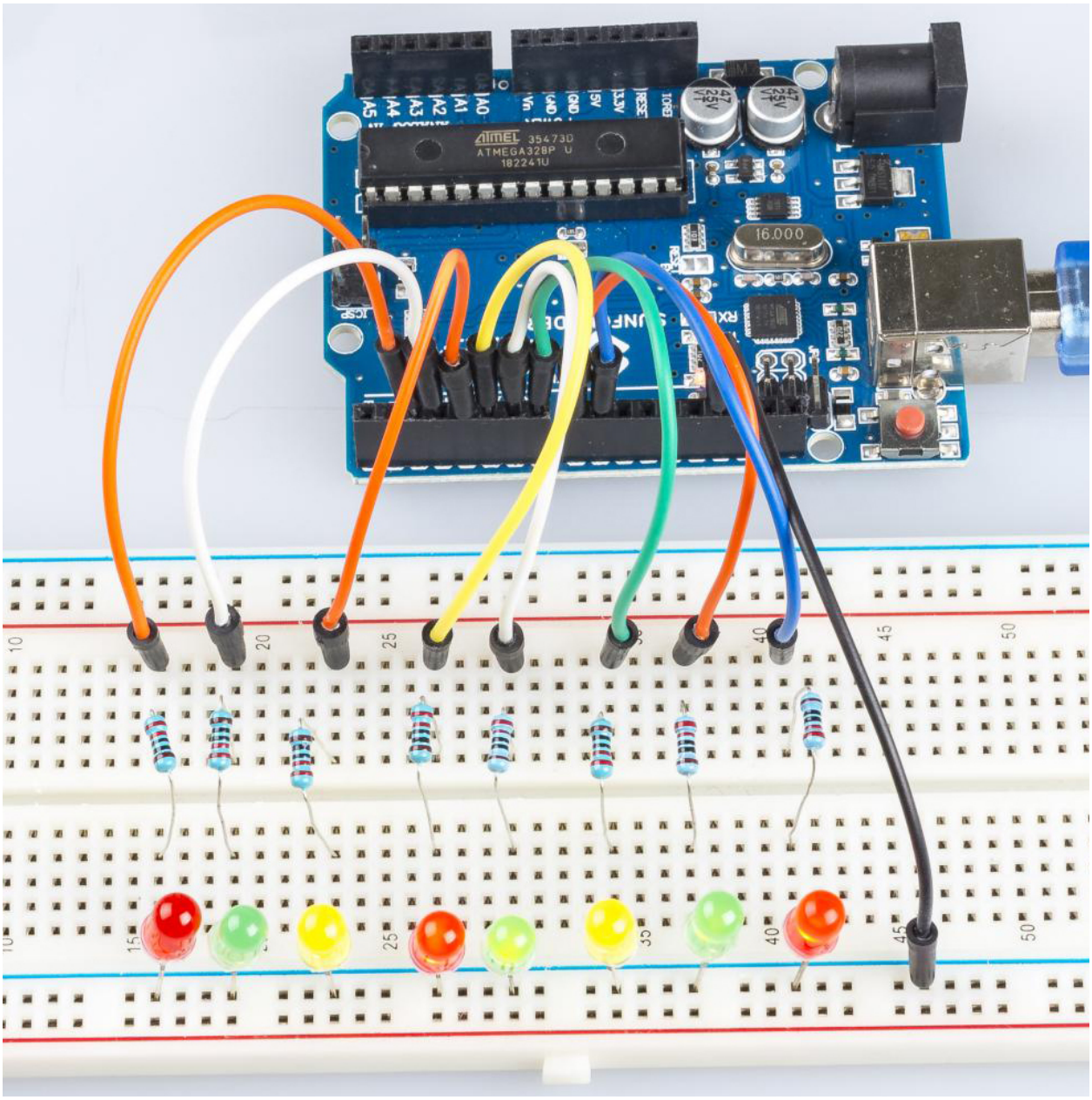
第 2 步：打开代码文件 `Lesson_2_Flowing_LED_Lights.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。



现在，你应该看到八个 LED 灯从连接在第 2 脚的 LED 到第 9 脚的 LED 逐一变亮，然后从第 9 脚的 LED 到第 2 脚的 LED 依次变暗。这整个过程将重复进行，直到电路断电。



6.2.5 代码

6.2.6 代码分析

for() 语句

```
void setup()
{
    //set pins 2 through 9 as output
    for (int i = 2; i <= 9; i++)
    {
        pinMode(i, OUTPUT); //initialize a as an output
```

(续下页)

(接上页)

```
    }
}
```

`for (initialization; condition; increment) { //statement(s); }`: for 语句用于重复一个大括号内的语句块。初始化首先发生，而且正好一次。每次通过循环时，都要对条件进行测试；如果条件为真，语句块和增量就会被执行，然后再次测试条件。当条件变为假时，循环结束。

设置流水灯

使用 `for()` 语句将 2 引脚 ~9 引脚设置为高电平。

```
for (int a = 2; a <= 9; a++)
{
    digitalWrite(a, HIGH); //turn this led on
    delay(100); //wait for 100 ms
}
```

然后让 8 个 LED 依次从 9 引脚到 2 引脚熄灭。

```
for (int a = 9; a <= 2; a--)
{
    digitalWrite(a, LOW); //turn this led on
    delay(100); //wait for 100 ms
}
```

最后用同样的方法将 9 引脚到 2 引脚的 8 个 LED 依次点亮，让它们依次熄灭。

```
for (int a = 9; a <= 2; a--)
{
    digitalWrite(a, HIGH); //turn this led on
    delay(100); //wait for 100 ms
}
for (int a = 2; a <= 9; a++)
{
    digitalWrite(a, LOW); //turn this led on
    delay(100); //wait for 100 ms
}
```

6.2.7 实验总结

通过这个实验，你已经学会了如何使用 `for()` 语句，当你想缩短代码时，这是一个非常有用的语句。

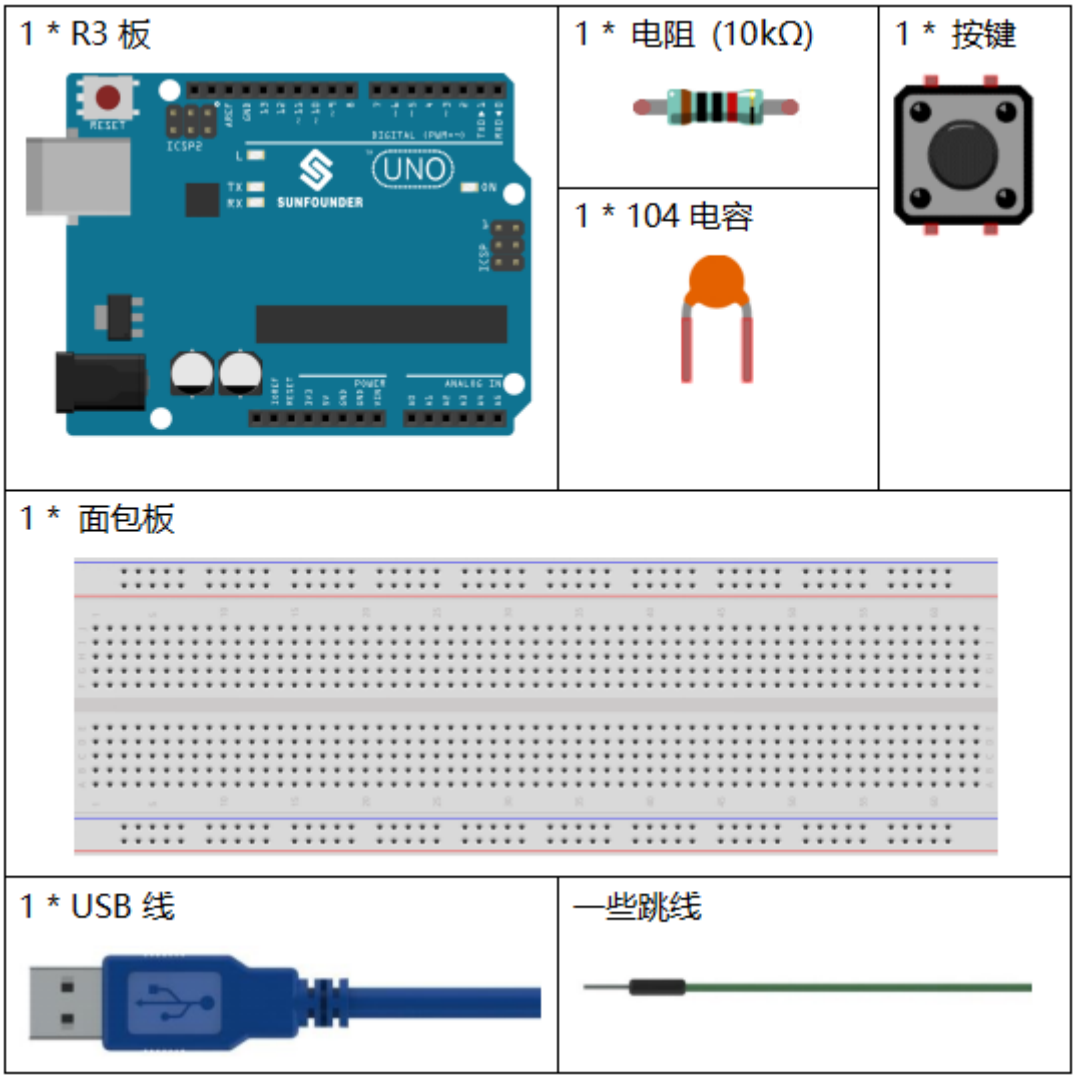
6.3 第 3 课按键

6.3.1 介绍

在本实验中，我们将学习如何使用 I/O 端口和按键来打开/关闭 LED。

“I/O 端口”是指 INPUT 和 OUTPUT 端口。这里使用控制板的 INPUT 端口来读取外部设备的输出。由于板子本身有一个 LED（连接到引脚 13），为了方便起见，你可以使用这个 LED 来做这个实验。

6.3.2 所需器件



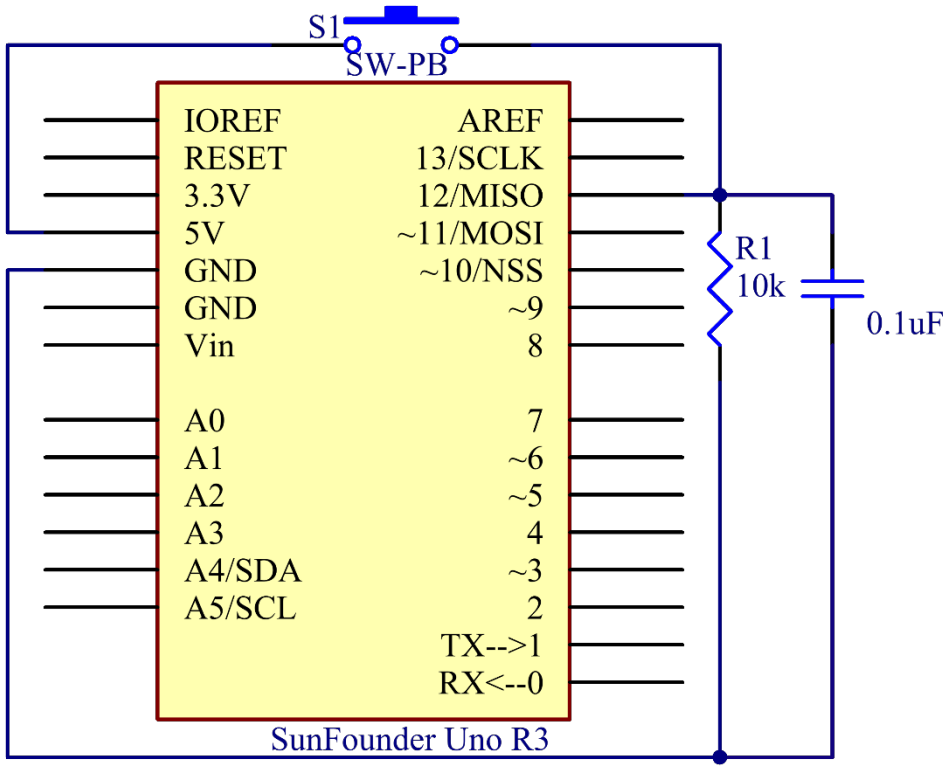
- *SunFounder R3 板*
- 面包板
- 跳线
- 电阻
- 电容
- 按键

6.3.3 原理图

按键一端接 12 脚，同时接下拉电阻和 0.1uF（104）电容（用来消除抖动让按键工作时能够输出稳定电平）。

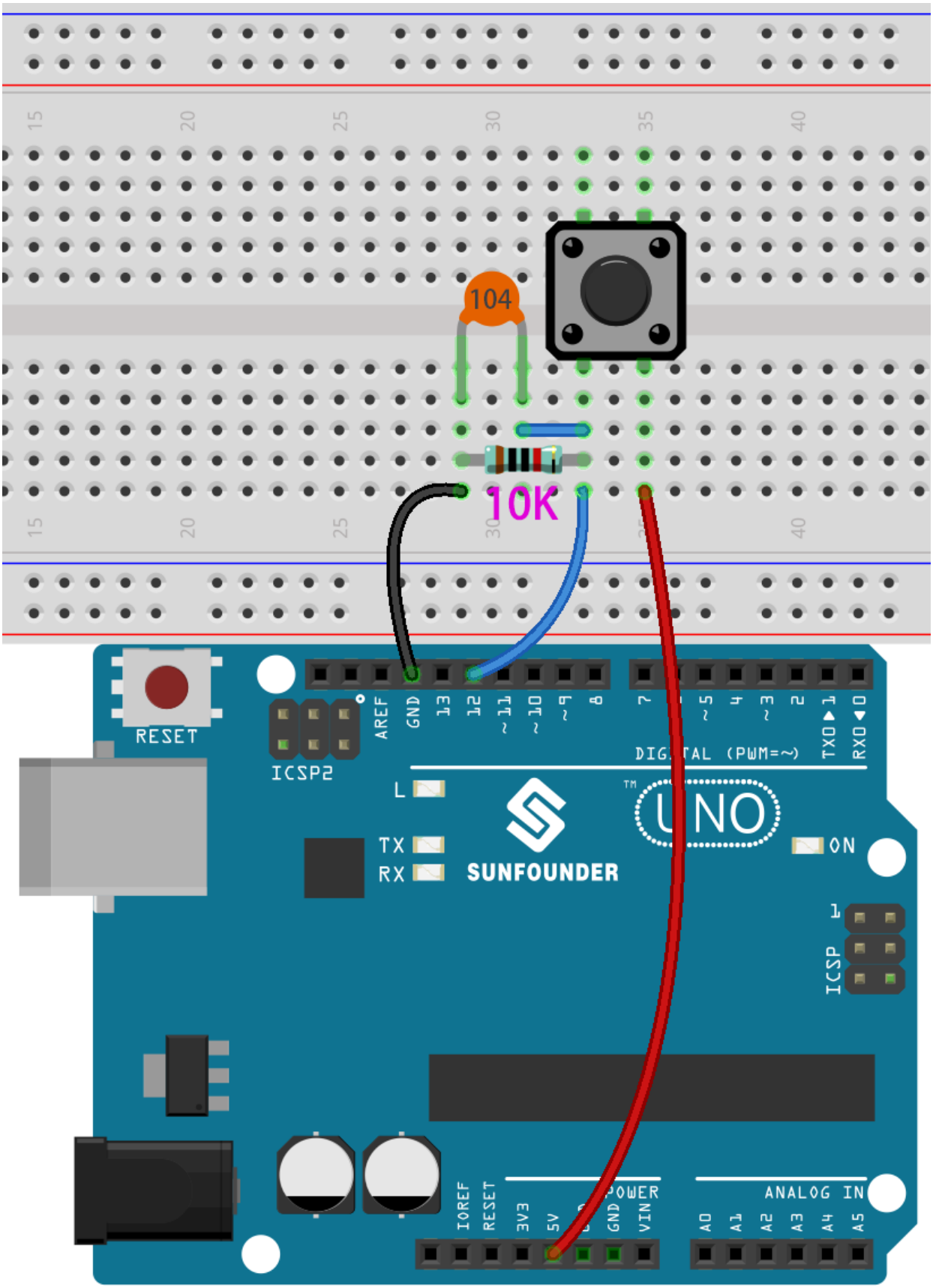
将电阻和电容的另一端连接到 GND，将按键另一端的一个引脚连接到 5V。按下按键时，引脚 12 为 5V（高电平），此时将 13 引脚设置为高电平来将控制板上的内置 LED 点亮。然后松开按键（引脚 12 变为低电平），引脚 13 为低电平。因此，我们将看到 LED 在按下和释放按键时交替亮起和熄灭。

原理图如下所示：



6.3.4 实验步骤

第 1 步：搭建电路。.

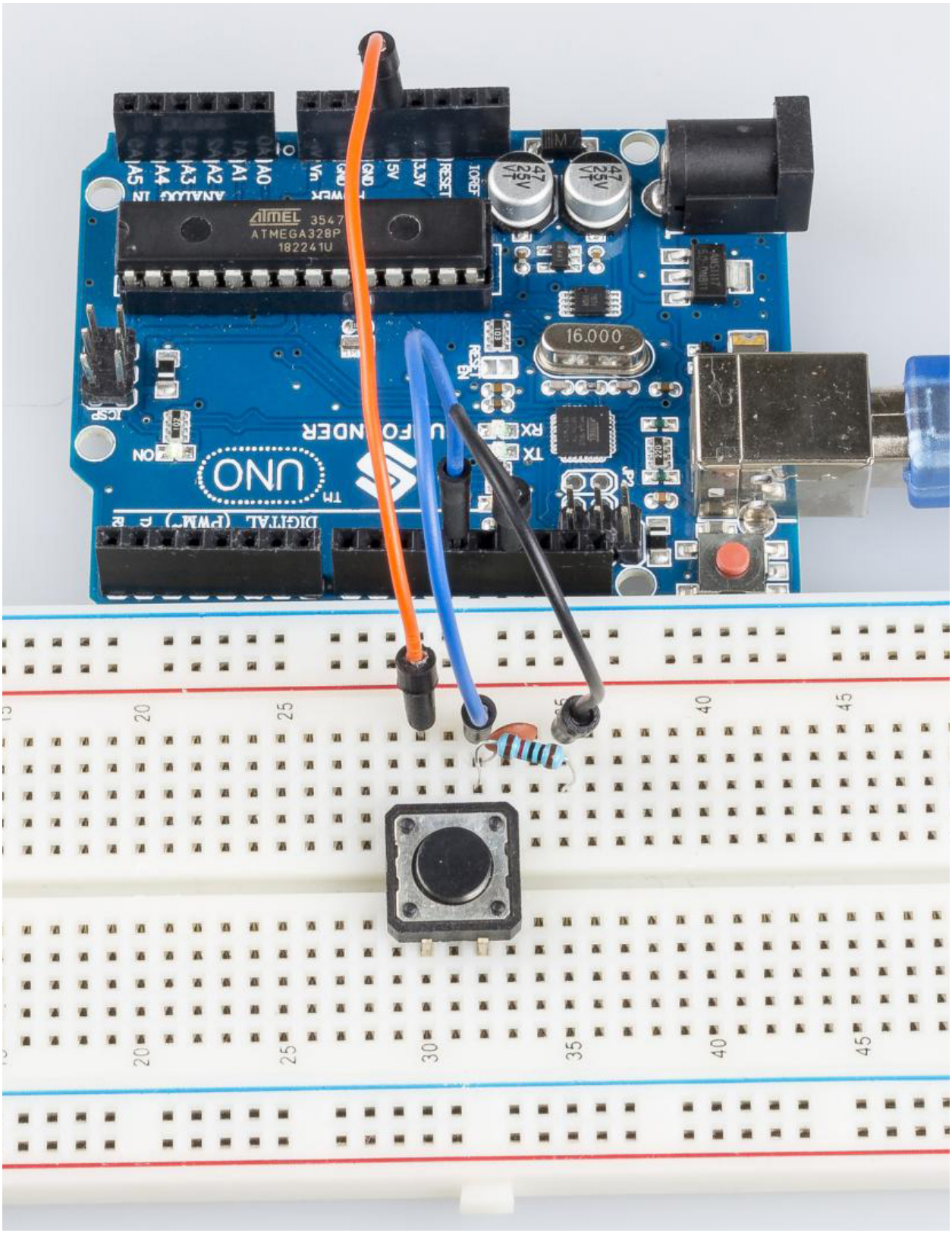

104
10K
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

第 2 步：打开代码文件 `Lesson_3_Button.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在按下按键，控制板上的 LED 被点亮。


ATMEL
ATMEGA328P
16.000
UNO

6.3.5 代码

6.3.6 代码分析

定义变量

```
const int buttonPin = 12; //the button connect to pin 12
const int ledPin = 13; //the led connect to pin13
int buttonState = 0; // variable for reading the pushbutton status
```

将按键连接到引脚 12，LED 已经连接到引脚 13。定义一个变量 buttonState 来存储按键的值。

设置引脚的输入输出状态

```
pinMode(buttonPin, INPUT); //initialize thebuttonPin as input
pinMode(ledPin, OUTPUT); //initialize the led pin as output
```

本次实验我们需要知道按键的状态，所以这里设置 buttonPin 为 INPUT ； 要设置 LED 的高/低，我们将 LedPin 设置为 OUTPUT。

读取按键状态

```
buttonState = digitalRead(buttonPin);
```

buttonPin``(Pin12) 是数字引脚；这里是读取按键的值并将其存储在 ``buttonState 中。

- digitalRead (Pin)：从指定的数字引脚读取值，无论是高电平还是低电平。

按键按下时让 LED 点亮

```
if (buttonState == HIGH )
{
    digitalWrite(ledPin, HIGH); //turn the led on
}
else
{
    digitalWrite(ledPin, LOW); //turn the led off
}
```

在这部分代码中，当 buttonState 为 HIGH 时，让 ledPin 为 HIGH ，LED 会被点亮。

由于按键的一端已连接至 5V，另一端已连接至引脚 12，因此按下按键时，引脚 12 为 5V（高电平）。然后用 if () 判断；如果条件为真，则 LED 将亮起。

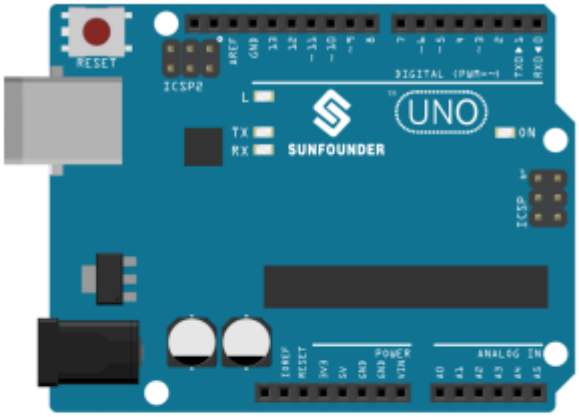
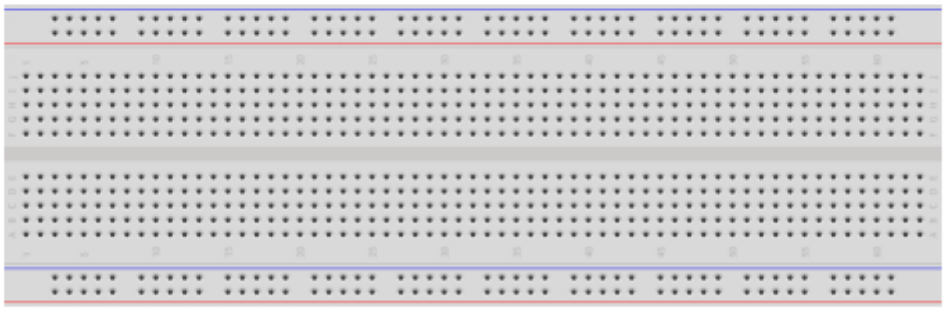
else 意味着当 if(conditional) 被确定为 false 时，运行 else。

6.4 第 4 课蜂鸣器

6.4.1 介绍

每当你想发出声音时，蜂鸣器都是你实验中的绝佳工具。在本课中，我们将学习如何驱动有源蜂鸣器来制作一个简单的门铃。

6.4.2 所需器件

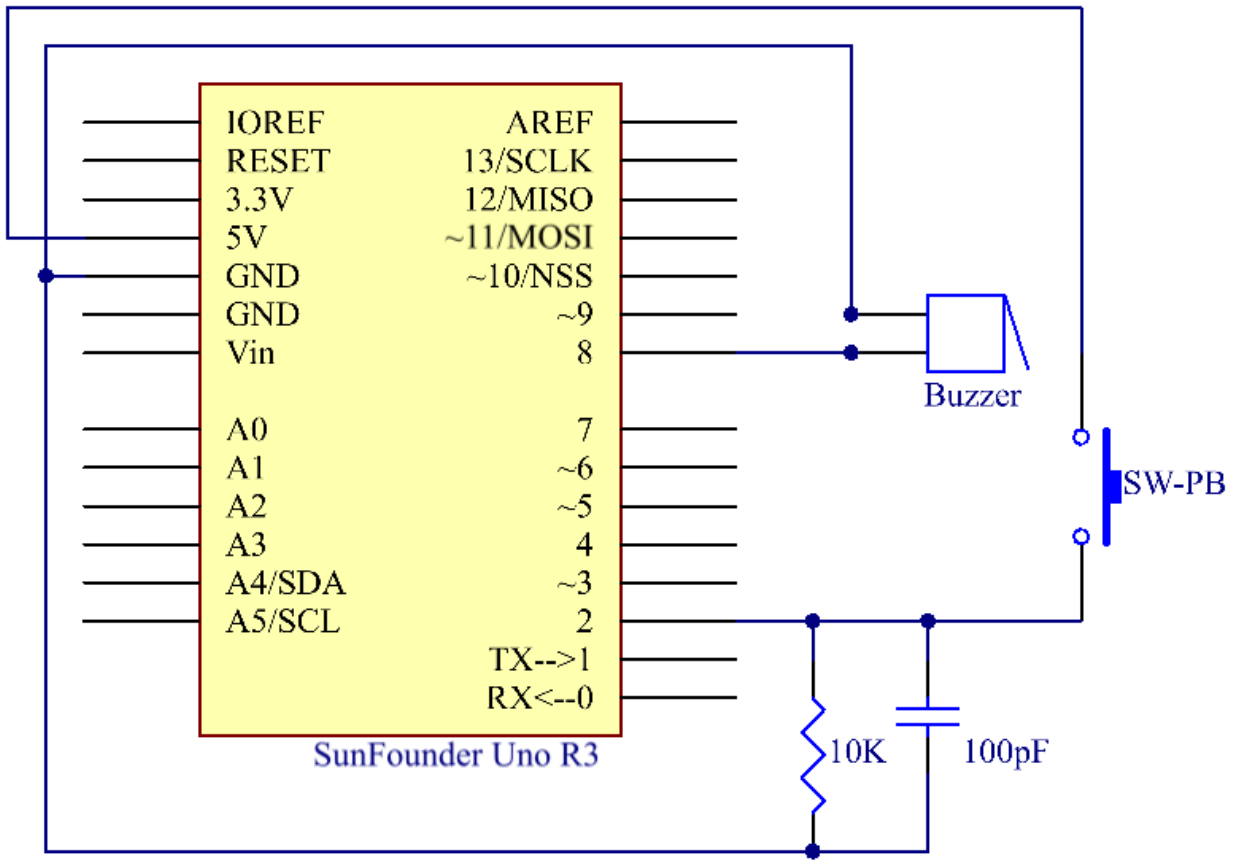
<table>
<tr><td>1 * R3 板
  </td><td>1 * 蜂鸣器 (有源)
  </td><td>1 * 104 电容
  </td></tr>
<tr><td colspan="2">1 * 按键
  </td><td>1 * 电阻 (10kΩ)
  </td></tr>
<tr><td colspan="3">1 * 面包板
  </td></tr>
<tr><td>1 * USB 线
  </td><td colspan="2">一些跳线
  </td></tr>
</table>

- *SunFounder R3 板*
- 面包板
- 跳线
- 电阻
- 电容
- 按键
- 蜂鸣器

6.4.3 原理图

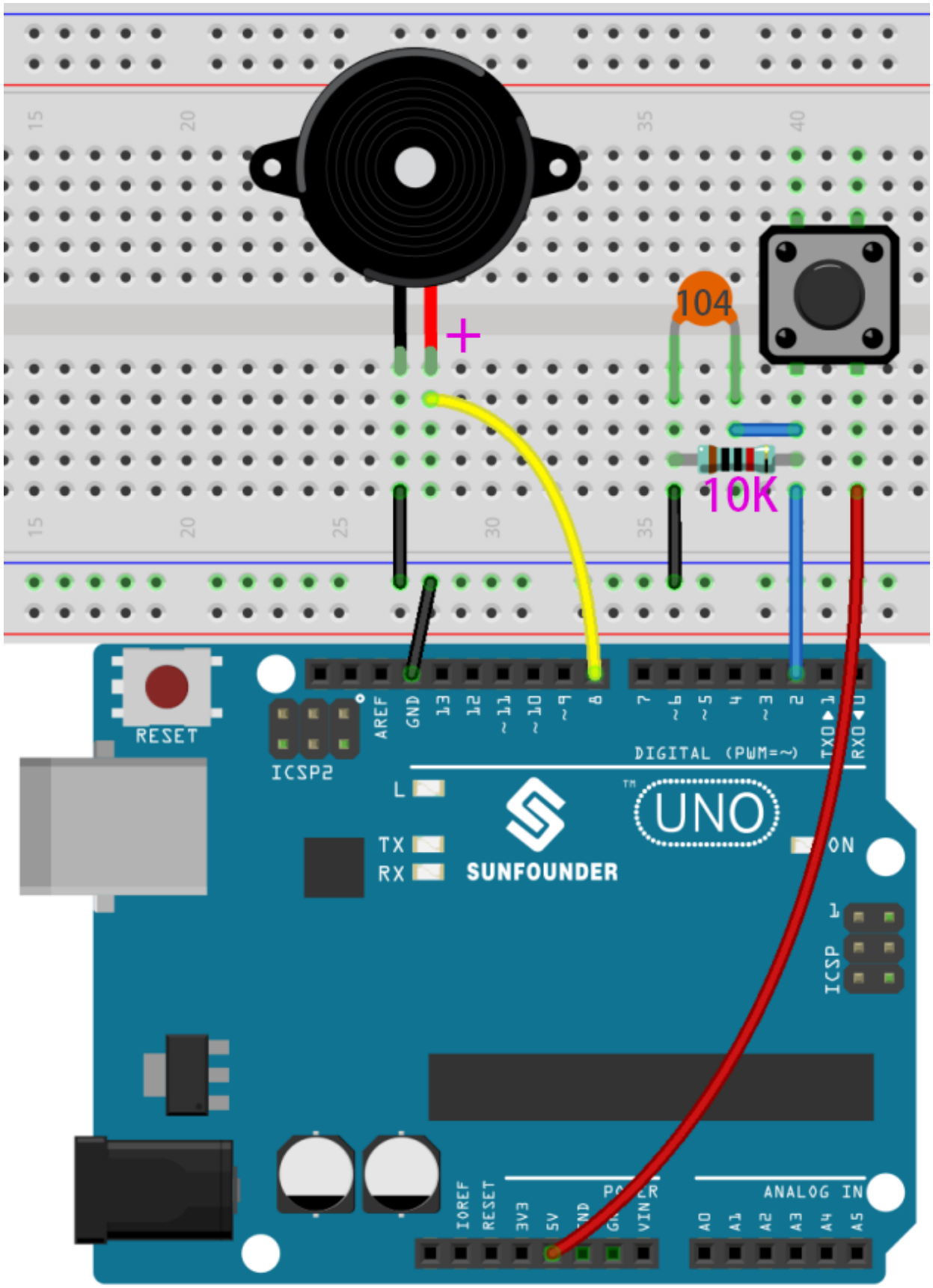
在这个课程中，使用的是有源蜂鸣器。

原理图如下所示：



6.4.4 实验步骤

第 1 步：搭建电路。（蜂鸣器长的引脚为阳极，短的为阴极）

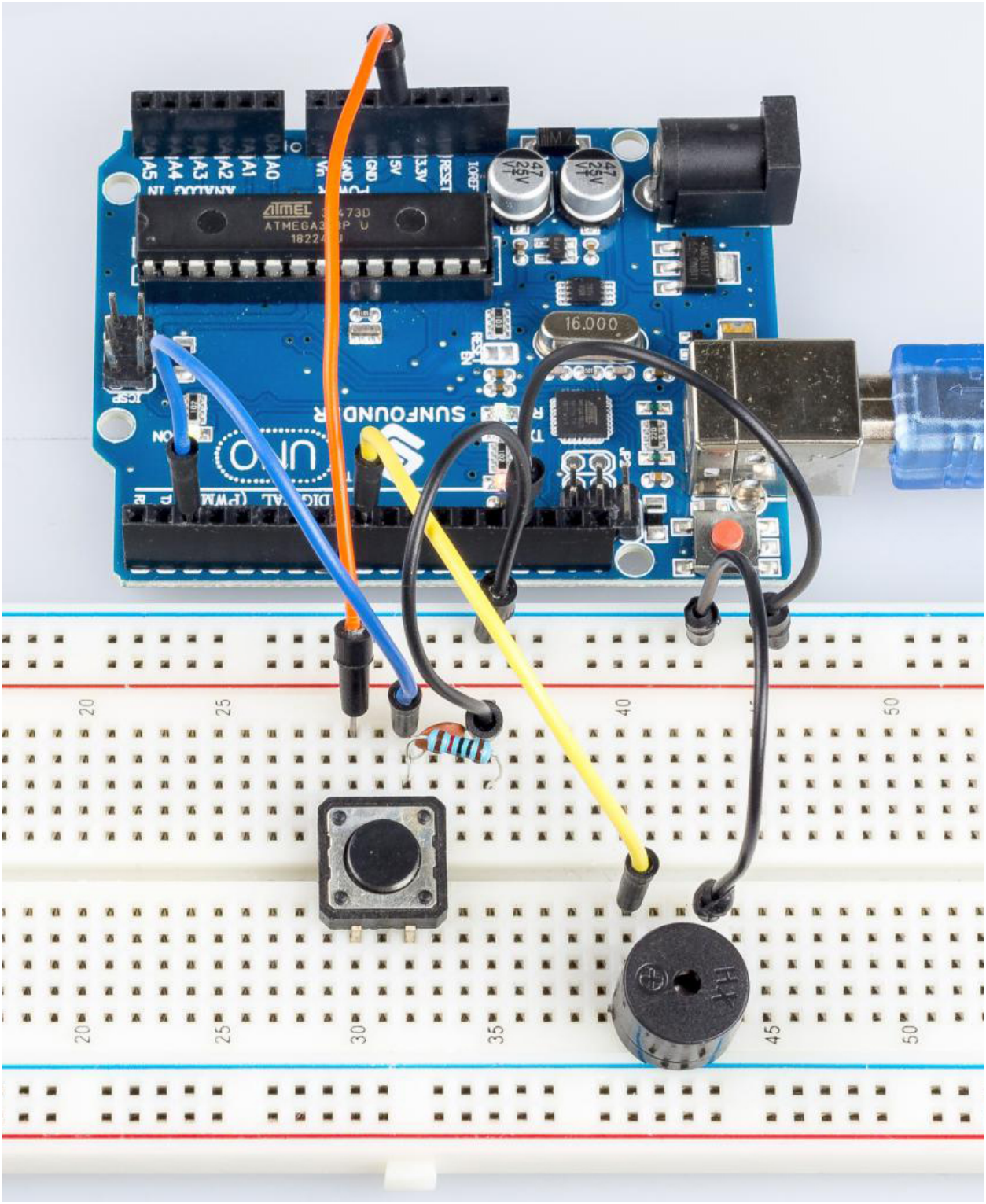

104
+
10K
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5

第 2 步：打开代码文件 `Lesson_4_Buzzer.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

按下按键，蜂鸣器将发出声音。



6.4.5 Code

6.4.6 代码分析

定义变量

```
const int buttonPin = 2; //the button connect to pin2
const int buzzerPin = 8; //the led connect to pin8
/**********************************/
int buttonState = 0; //variable for reading the pushbutton status
```

将按键连接到引脚 2，将蜂鸣器连接到引脚 8。定义一个变量 buttonState 来存储按键的值。

设置引脚的输入输出状态

```
void setup()
{
    pinMode(buttonPin, INPUT); //initialize the buttonPin as input
    pinMode(buzzerPin, OUTPUT); //initialize the buzzerpin as output
}
```

本次实验我们需要知道按键的状态，所以这里设置 buttonPin 为 INPUT；要设置蜂鸣器的高/低，我们将 buzzerPin 设置为 OUTPUT。

读取按钮状态

```
buttonState = digitalRead(buttonPin);
```

buttonPin (Pin2) 是数字引脚；这里是读取按钮的值并将其存储在 buttonState 中。

- digitalRead (Pin)：从指定的数字引脚读取值，无论是高电平还是低电平。

按下按钮让蜂鸣器发出声音

```
if (buttonState == HIGH ) //When press the button, run the following code.
{
    for (i = 0; i < 50; i++)
    /*When i=o, which accords with the condition i<=50, i++ equals to 1
    (here in i = i + 1, the two "i"s are not the same, but i(now)  = i (before)  + 1).
    Run the code in the curly braces: let the buzzer beep for 3ms and stop for 3ms.
    Then repeat 50 times.*/

    {
        digitalWrite(buzzerPin, HIGH); //Let the buzzer beep.
        delay(3); //wait for 3ms
        digitalWrite(buzzerPin, LOW); //Stop the buzzer.
        delay(3); //wait for 3ms
    }

    for (i = 0; i < 80; i++) //Let the buzzer beep for 5ms and stop for 5ms, repeat␣
↪80 times.
    {
        digitalWrite(buzzerPin, HIGH);
        delay(5); //wait for 5ms
        digitalWrite(buzzerPin, LOW);
        delay(5); //wait for 5ms
    }
}
```

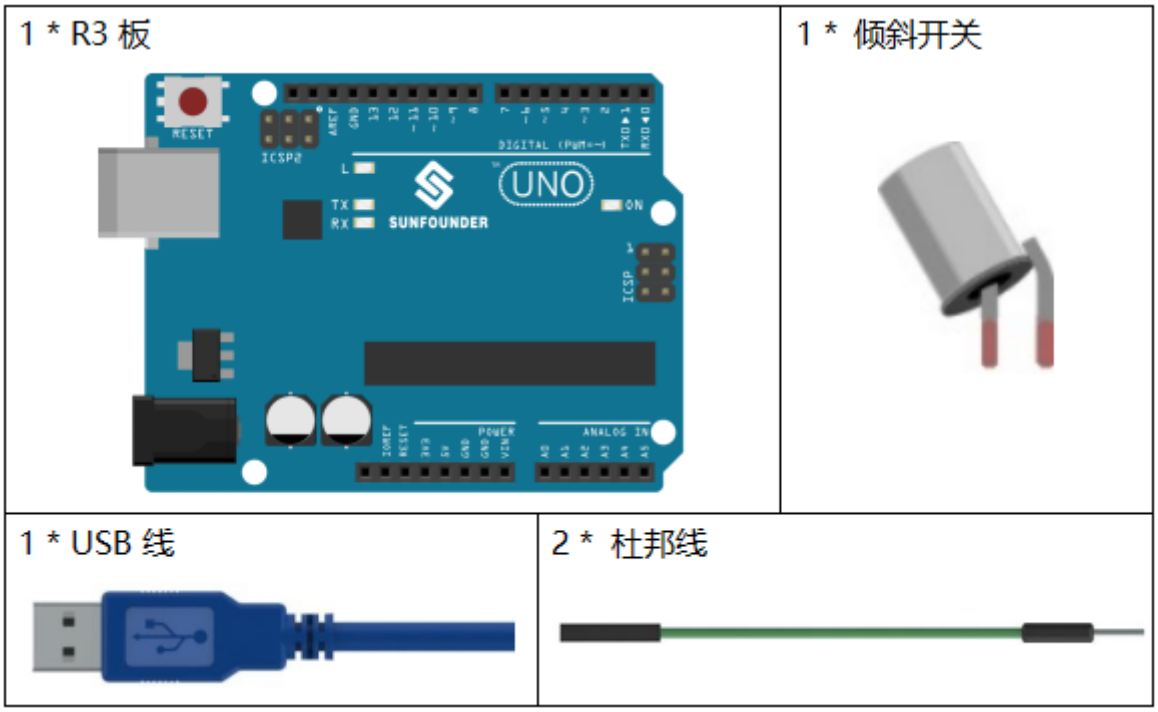
在这部分，当 `buttonState` 为高电平时，让蜂鸣器以不同的频率发出哔哔声，可以模拟门铃。

6.5 第 5 课倾斜开关

6.5.1 介绍

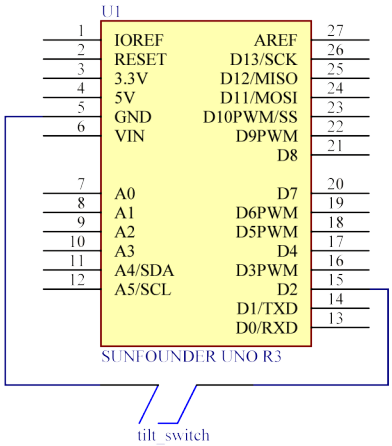
这里用的倾斜开关是内部有一个金属球，用来检测小角度的倾斜值。

6.5.2 所需器件



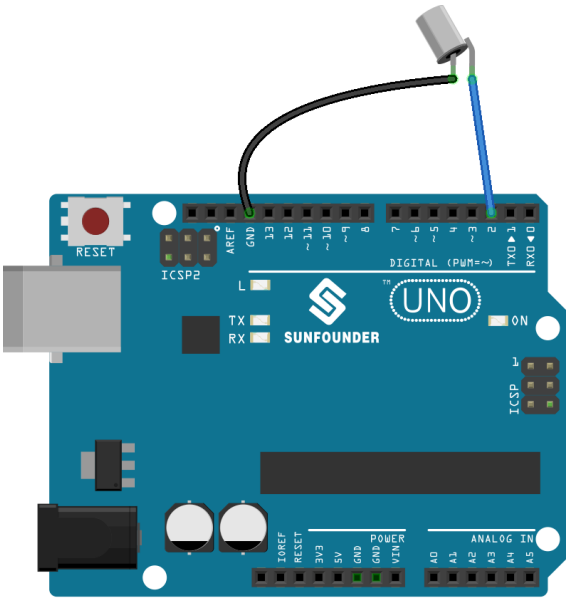
- *SunFounder R3 板*
- 面包板
- 跳线
- 倾斜开关

6.5.3 原理图



6.5.4 实验步骤

第 1 步：搭建电路。.

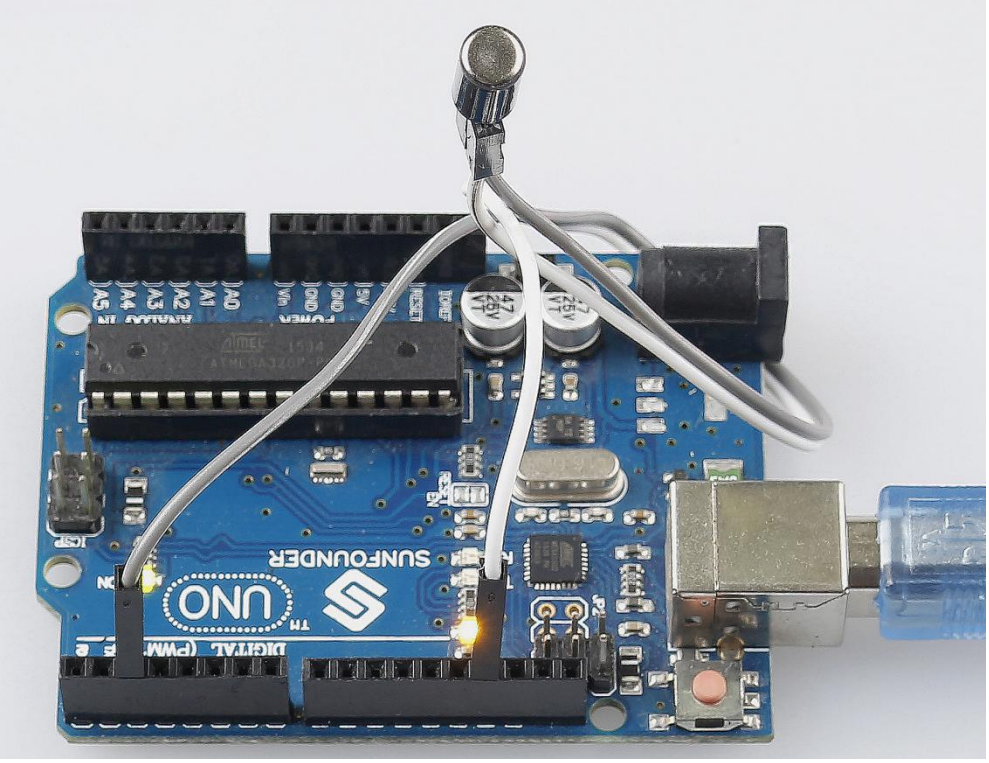


第 2 步：打开代码文件 `Lesson_5_Tilt_Switch.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，将开关倾斜，控制板上的 LED 将会被点亮。



6.5.5 代码

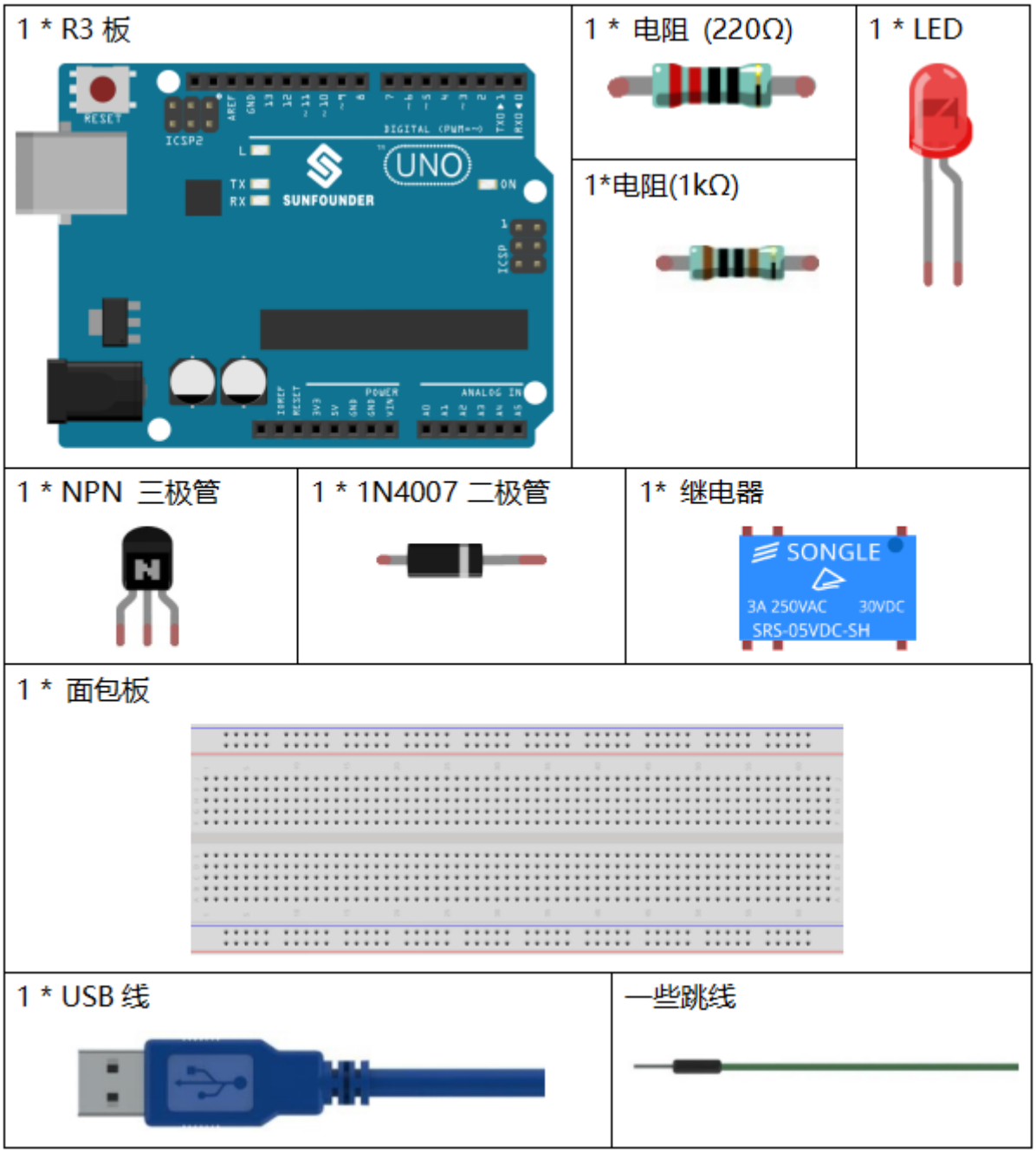
整个代码很简单，倾斜开关的一个脚接 pin2，另一脚接 GND，当倾斜开关时，开关的两个脚接 GND，然后让 pin13 上的 LED 点亮。

6.6 第 6 课继电器

6.6.1 介绍

我们可能知道，继电器是一种设备，用于响应所施加的输入信号在两个或多个点或设备之间提供连接。换句话说，继电器在控制器和设备之间提供隔离，因为设备可以在交流电和直流电下工作。然而，它们从工作于直流的微控制器接收信号，因此需要继电器来弥补差距。当你需要用小电信号控制大量电流或电压时，继电器非常有用。

6.6.2 所需器件



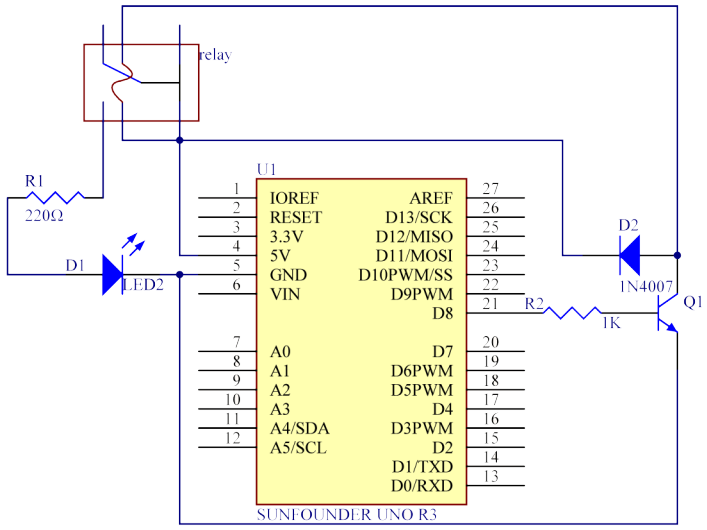
- *SunFounder R3 板*
- 面包板
- 跳线
- 电阻
- 继电器
- 三极管

- 二极管
- *LED* 发光二极管

6.6.3 原理图

将 1K 电阻（晶体管通电时限流）连接到控制板的第 8 脚，然后连接到 NPN 晶体管，其集电极连接到继电器的线圈和发射极到 GND；将继电器的常开触点连接到 LED，然后连接到 GND。因此，当高电平信号提供给引脚 8 时，晶体管被激励，从而使继电器的线圈导通。然后它的常开触点闭合，LED 将亮起。当引脚 8 处于低电平时，LED 将熄灭。

原理图如下所示：



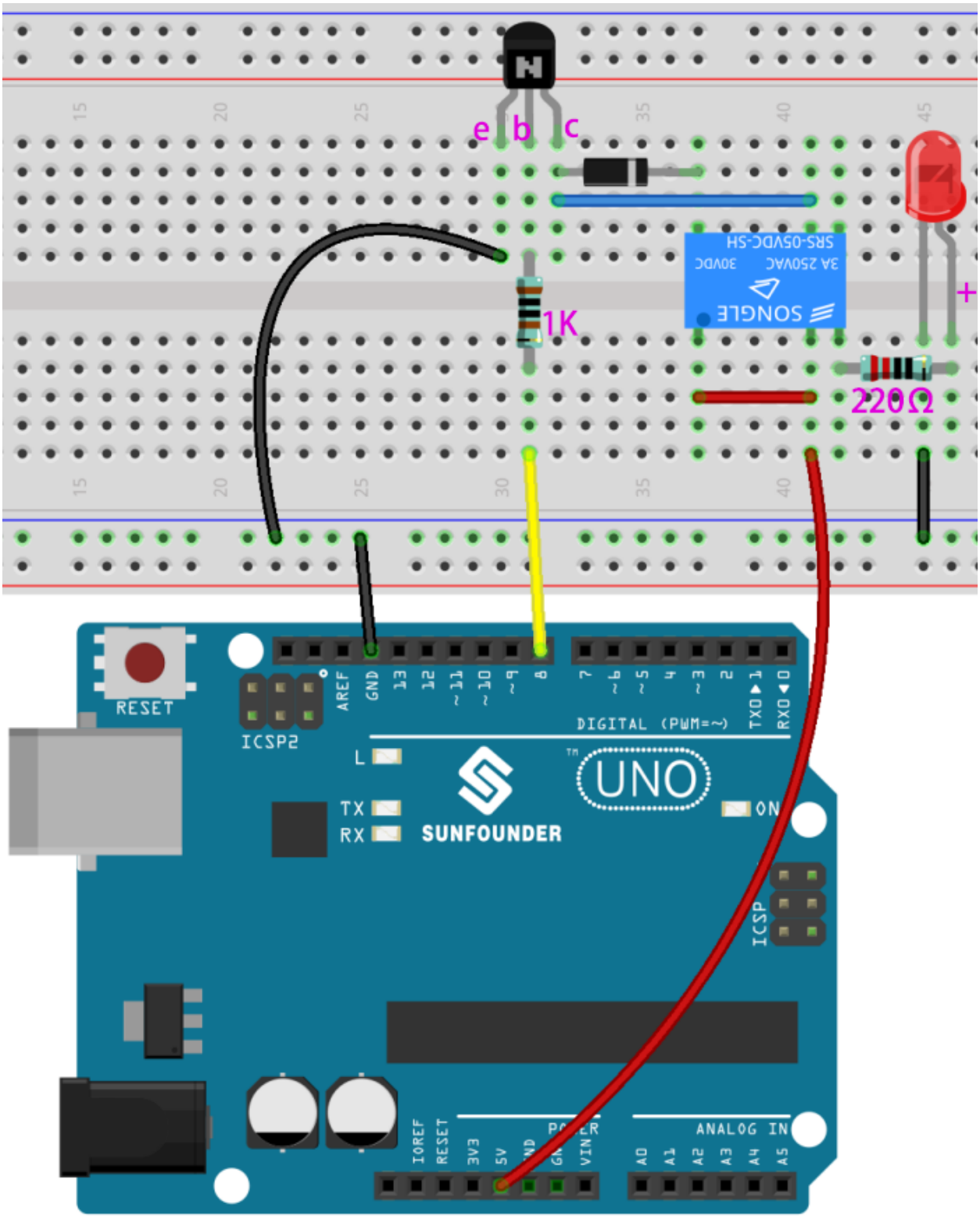
整流二极管的作用

当电压输入由 High（5V）变为 Low（0V）时，晶体管由饱和（放大、饱和、截止三种工作状态）变为截止，线圈中的电流突然无法流过。此时，如果没有整流二极管，线圈两端会产生反电动势（EMF），底部为正极，顶部为负极，电压高于 100V。这个电压加上来自晶体管电源的电压大到足以烧毁它。因此，整流二极管在将这个反电动势按上图箭头方向放电时极为重要，因此晶体管对 GND 的电压不高于 +5V（+0.7V）。

在本实验中，当继电器闭合时，LED 会亮起；当继电器打开时，LED 将熄灭。

6.6.4 实验步骤

第 1 步：搭建电路。

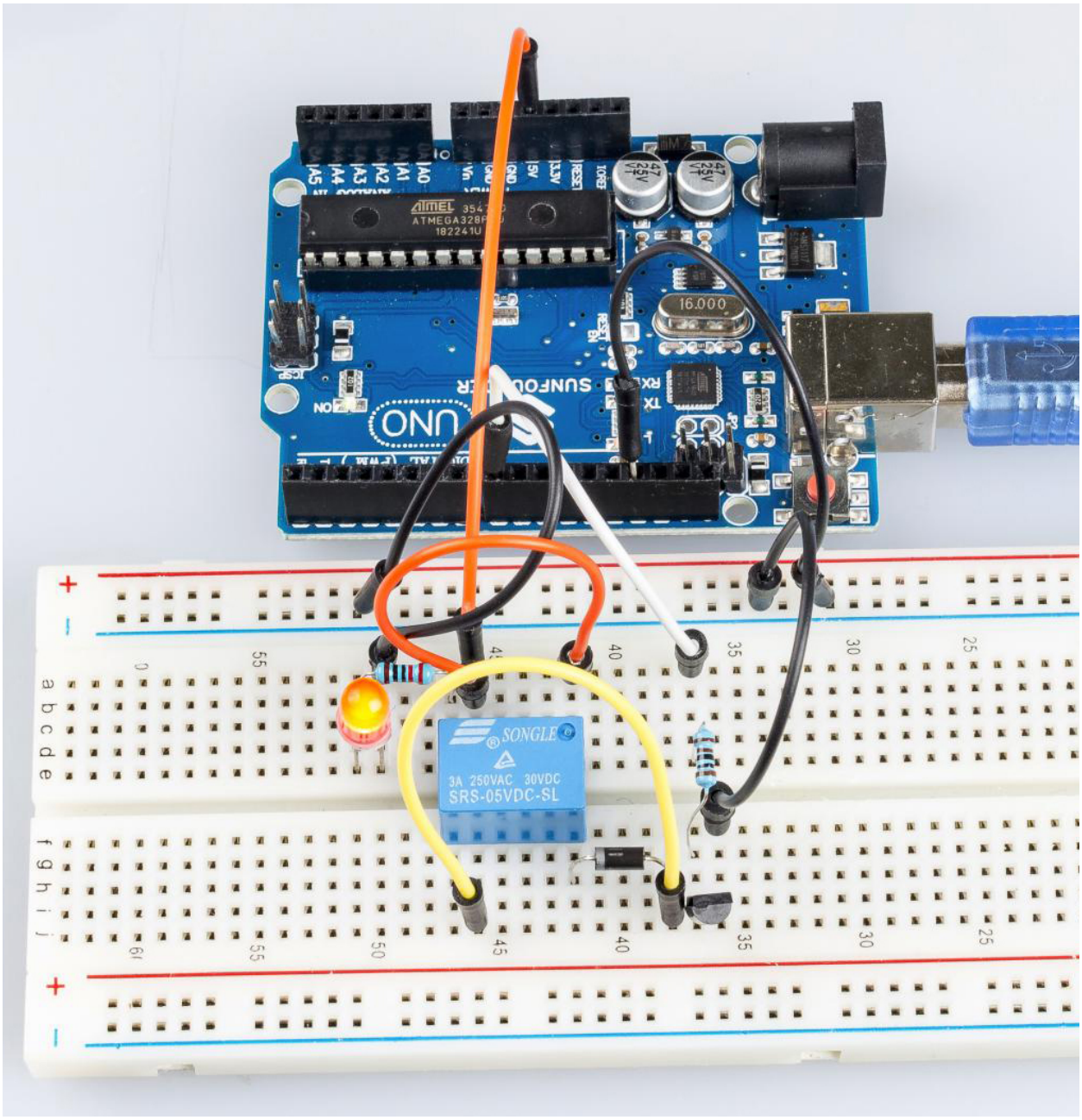


第 2 步：打开代码文件 `Lesson_6_Relay.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，发送一个高电平信号，继电器将关闭，LED 将亮起；发送一个低电平，它将打开并且 LED 将熄灭。此外，你还可以听到断开常闭触点并关闭常开触点引起的滴答声。



6.6.5 代码

6.6.6 代码分析

```
void loop()
{
    digitalWrite(relayPin, HIGH); //drive relay closure conduction
    delay(1000); //wait for a second
    digitalWrite(relayPin, LOW); //drive the relay is closed off
    delay(1000); //wait for a second
}
```

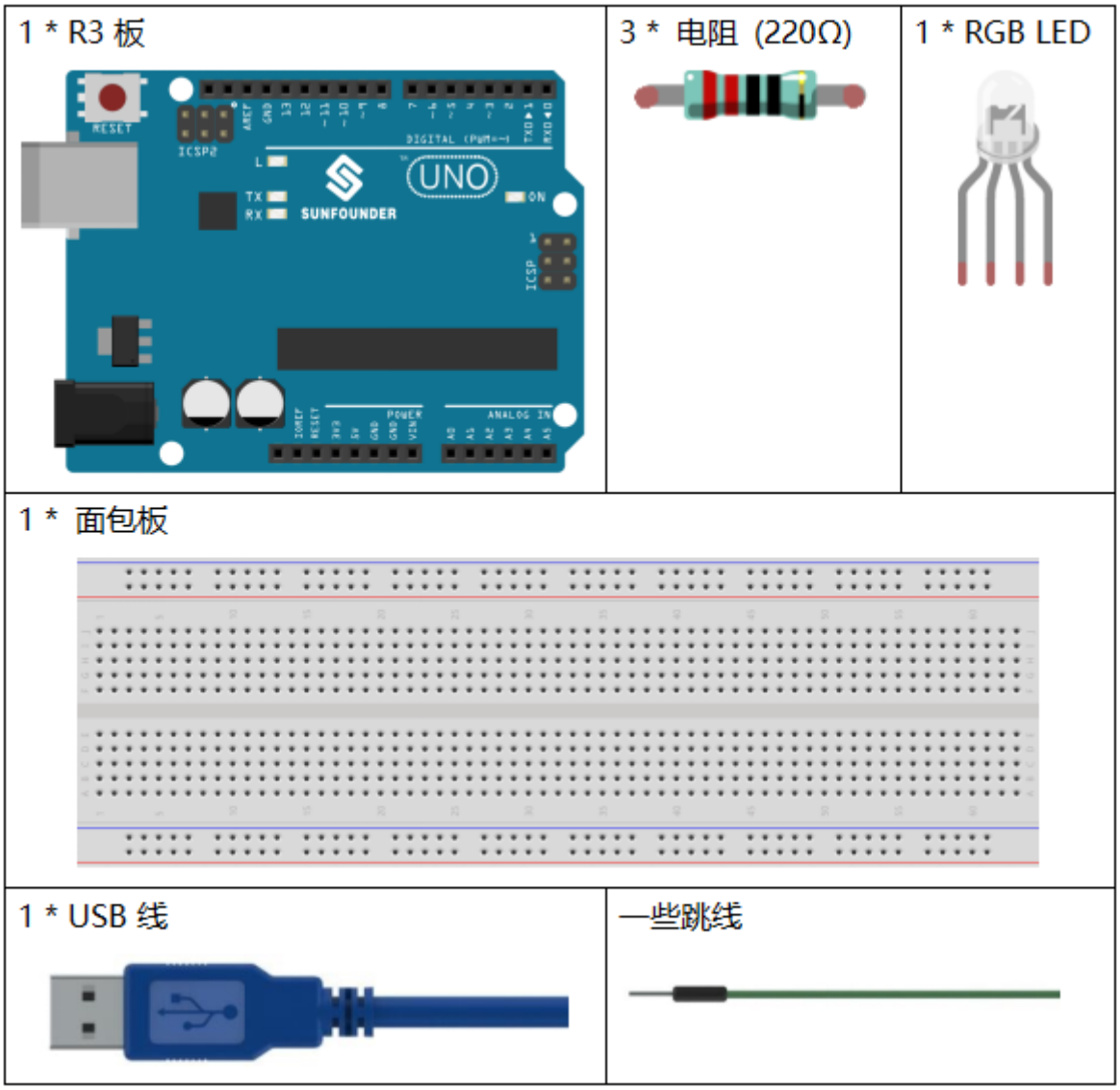
本实验中的代码很简单。首先，将 `relayPin` 设置为 `HIGH` 电平，连接到继电器的 LED 将亮起。然后将 `relayPin` 设置为低电平，LED 熄灭。

6.7 第 7 课 RGB LED

6.7.1 介绍

以前我们使用数字引脚来控制 LED 的亮度和亮度。在本课中，我们将使用 PWM 来控制 RGB LED 闪烁各种颜色。当 LED 的 R、G、B 引脚设置不同的 PWM 值时，其亮度会有所不同。当三种不同的颜色混合时，我们可以看到 RGB LED 闪烁不同的颜色。

6.7.2 所需器件

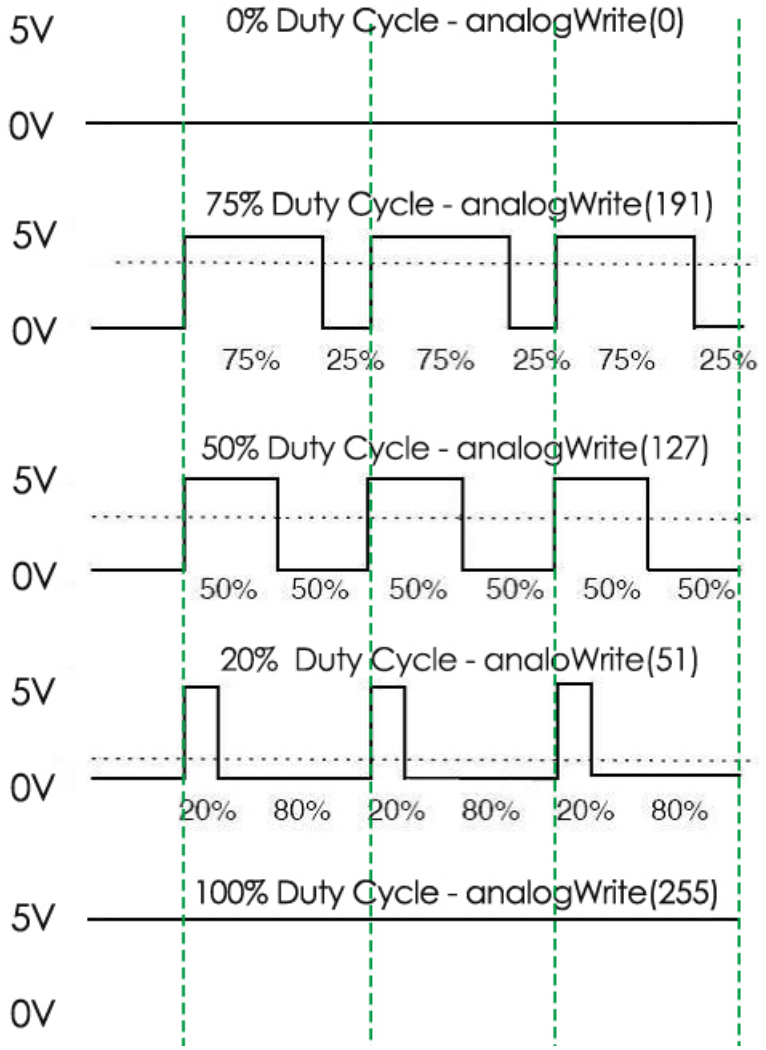


- *SunFounder R3 板*
- 面包板
- 跳线
- 电阻
- *RGB LED*

6.7.3 PWM

脉宽调制或 PWM 是一种通过数字方式获得模拟结果的技术。数字控制用于创建方波，这是一种在开和关之间切换的信号。这种开关模式可以通过改变信号打开的时间部分与信号关闭的时间来模拟完全打开（5 伏）和关闭（0 伏）之间的电压。“导通时间”的持续时间称为脉冲宽度。要获得不同的模拟值，你可以更改或调制该宽度。如果你使用某些设备（例如 LED）足够快地重复这种开关模式，它会是这样的：信号是 0 到 5V 之间的稳定电压，用于控制 LED 的亮度。

在下图中，绿线代表一个固定的时间段。该持续时间或周期是 PWM 频率的倒数。换句话说，当 Arduino 的 PWM 频率约为 500Hz 时，每条绿线的测量值为 2 毫秒。



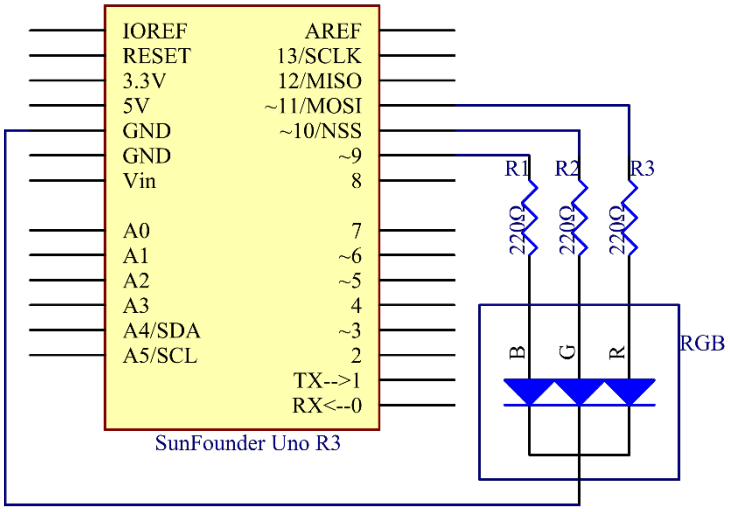
对 analogWrite() 的调用范围为 0 - 255，这样 `analogWrite(255)` 需要一个 100% 占空比（始终打开），`analogWrite(127)` 是 50% 占空比（一半时间)。

你会发现 PWM 值越小，转换成电压后的值越小。然后 LED 相应地变暗。因此，我们可以通过控制 PWM 值来控制 LED 的亮度。

6.7.4 原理图

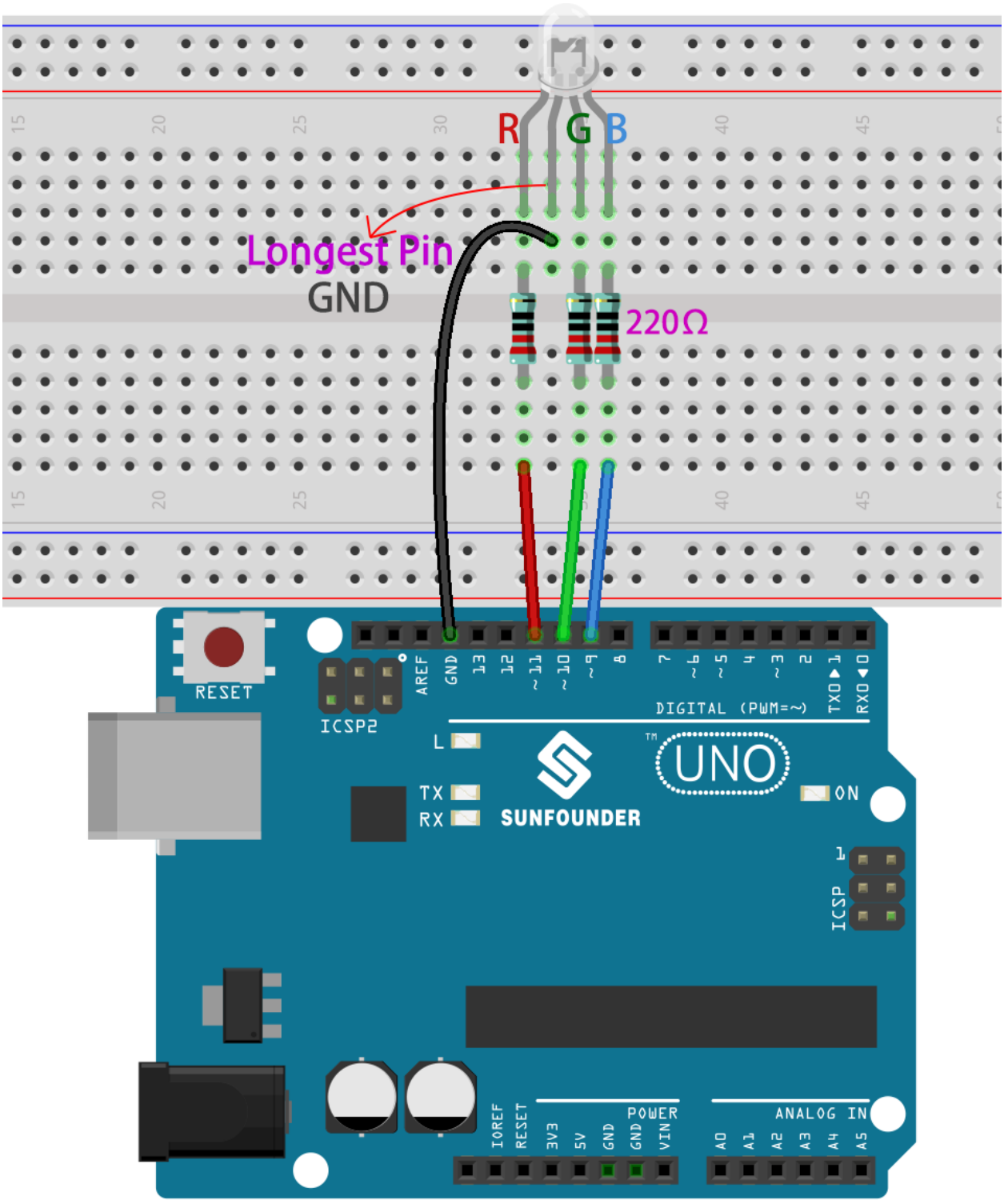
在 R3 板上，3，5，6 和 9~11 是 PWM 引脚，通过 `analogWrite()` 函数提供 8 位 PWM 输出。你可以连接这些引脚中的任何一个。这里我们将 0 到 255 之间的值输入到 RGB LED 的三个引脚，使其显示不同的颜色。R、G、B 引脚接限流电阻后，分别接 9、10、11 脚。LED 最长的引脚（GND）连接到控制板的 GND。当三个引脚被赋予不同的 PWM 值时，RGB LED 将显示不同的颜色。

原理图如下所示：



6.7.5 实验步骤

第 1 步：搭建电路。.



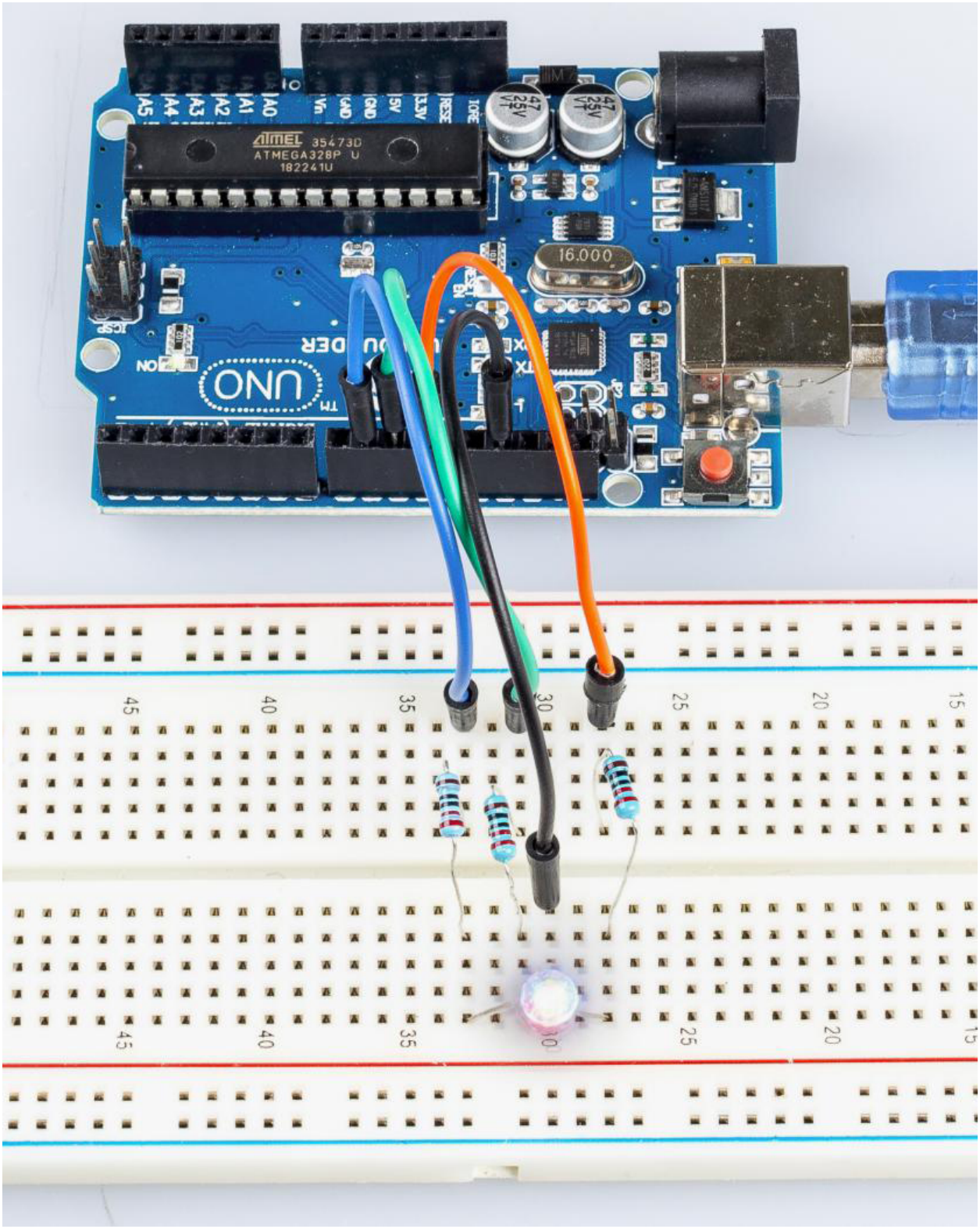
第 2 步：打开代码文件 `Lesson_7_RGB_LED.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

在这里你应该看到 RGB LED 首先循环闪烁红色、绿色和蓝色，然后是红色、橙色、黄色、绿色、蓝色、靛

蓝色和紫色。



6.7.6 代码

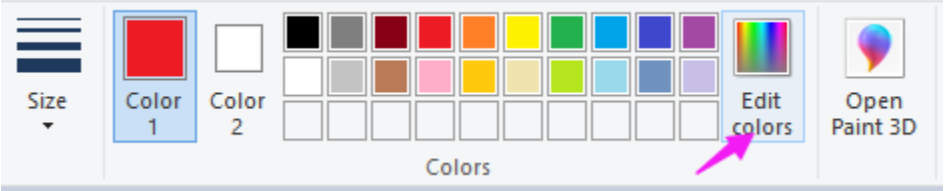
6.7.7 代码分析

设置颜色

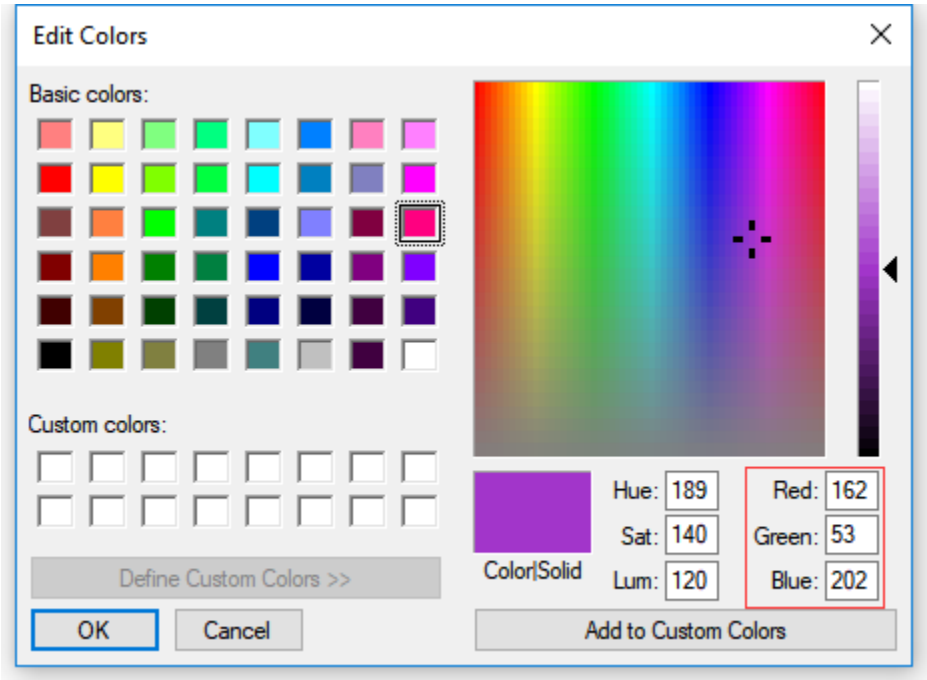
这里使用 `color()` 函数来设置 RGB LED 的颜色。在代码中，它被设置为闪烁 7 种不同的颜色。

你可以使用计算机上的绘图工具获取 RGB 值。

1. 打开计算机上的绘画工具，然后单击编辑颜色。



2. 选择一种颜色，即可看到该颜色的 RGB 值。在代码中填写它们。




```
void loop()  // run over and over again
{
    // Basic colors:
    color(255, 0, 0); // turn the RGB LED red
    delay(1000); // delay for 1 second
    color(0,255, 0); // turn the RGB LED green
    delay(1000); // delay for 1 second
    color(0, 0, 255); // turn the RGB LED blue
    delay(1000); // delay for 1 second
    // Example blended colors:
    color(255,0,252); // turn the RGB LED red
    delay(1000); // delay for 1 second
    color(237,109,0); // turn the RGB LED orange
```

(续下页)

(接上页)

```
    delay(1000); // delay for 1 second
    color(255,215,0); // turn the RGB LED yellow
    delay(1000); // delay for 1 second
    color(34,139,34); // turn the RGB LED green
    delay(1000); // delay for 1 second
    color(0,112,255); // turn the RGB LED blue
    delay(1000); // delay for 1 second
    color(0,46,90); // turn the RGB LED  indigo
    delay(1000); // delay for 1 second
    color(128,0,128); // turn the RGB LED purple
    delay(1000); // delay for 1 second
}
```

color() 函数

```
void color (unsigned char red, unsigned char green, unsigned char blue)// the color␣
↪generating function
{
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}
```

定义三个无符号字符变量，红色、绿色和蓝色。将它们的值写入 `redPin`、`greenPin` 和 `bluePin`。例如，颜色（128,0,128）是写 128 到 redPin，0 至 greenPin 和 128 至 bluePin。然后结果是 LED 闪烁紫色。

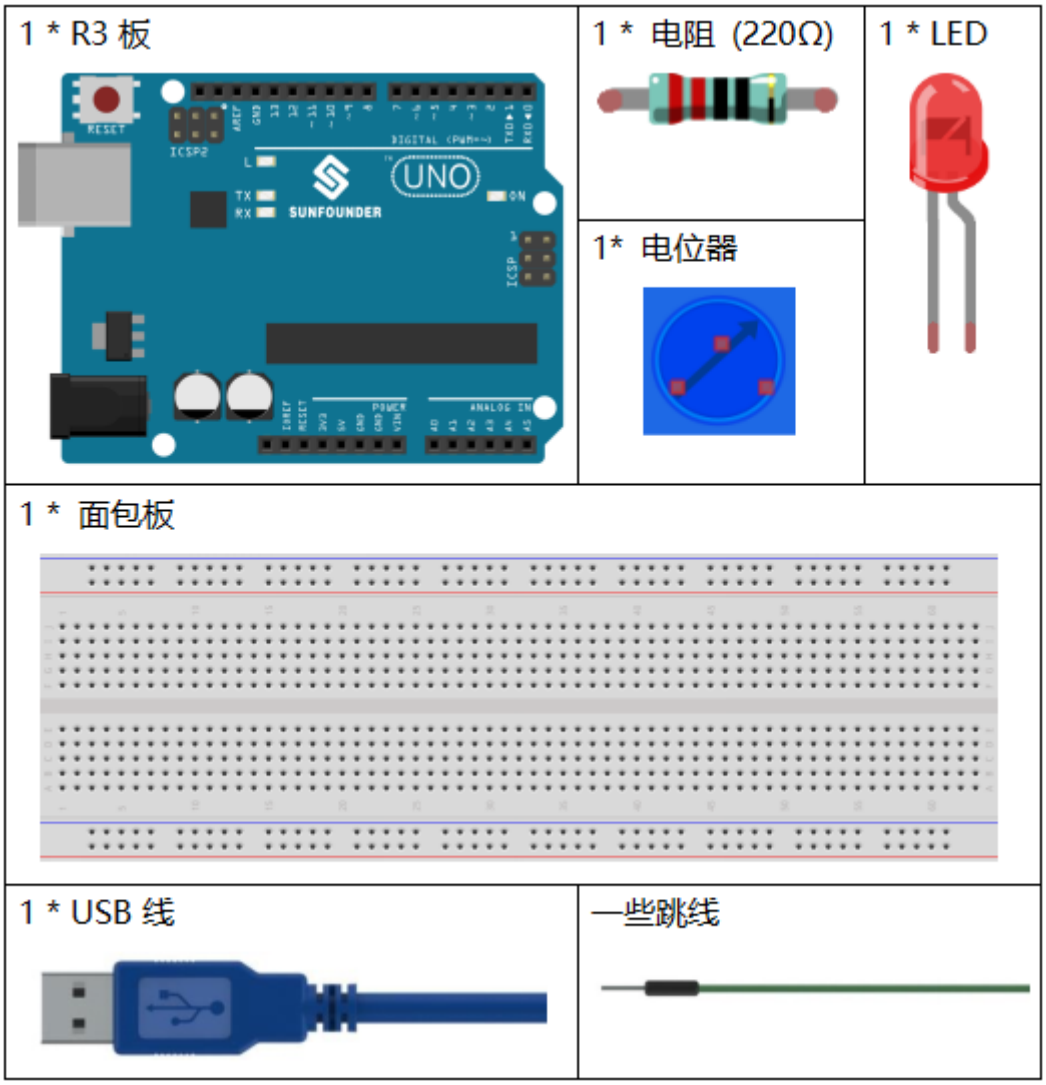
- `analogWrite()`：将模拟值（PWM 波）写入引脚。它与模拟引脚无关，仅适用于 PWM 引脚。在调用 `analogWrite()` 之前，你不需要调用 `pinMode()` 将引脚设置为输出。

6.8 第 8 课电位器

6.8.1 介绍

在本课中，让我们看看如何通过电位器改变 LED 的亮度，并在串口监视器中接收电位器的数据以查看其值变化。

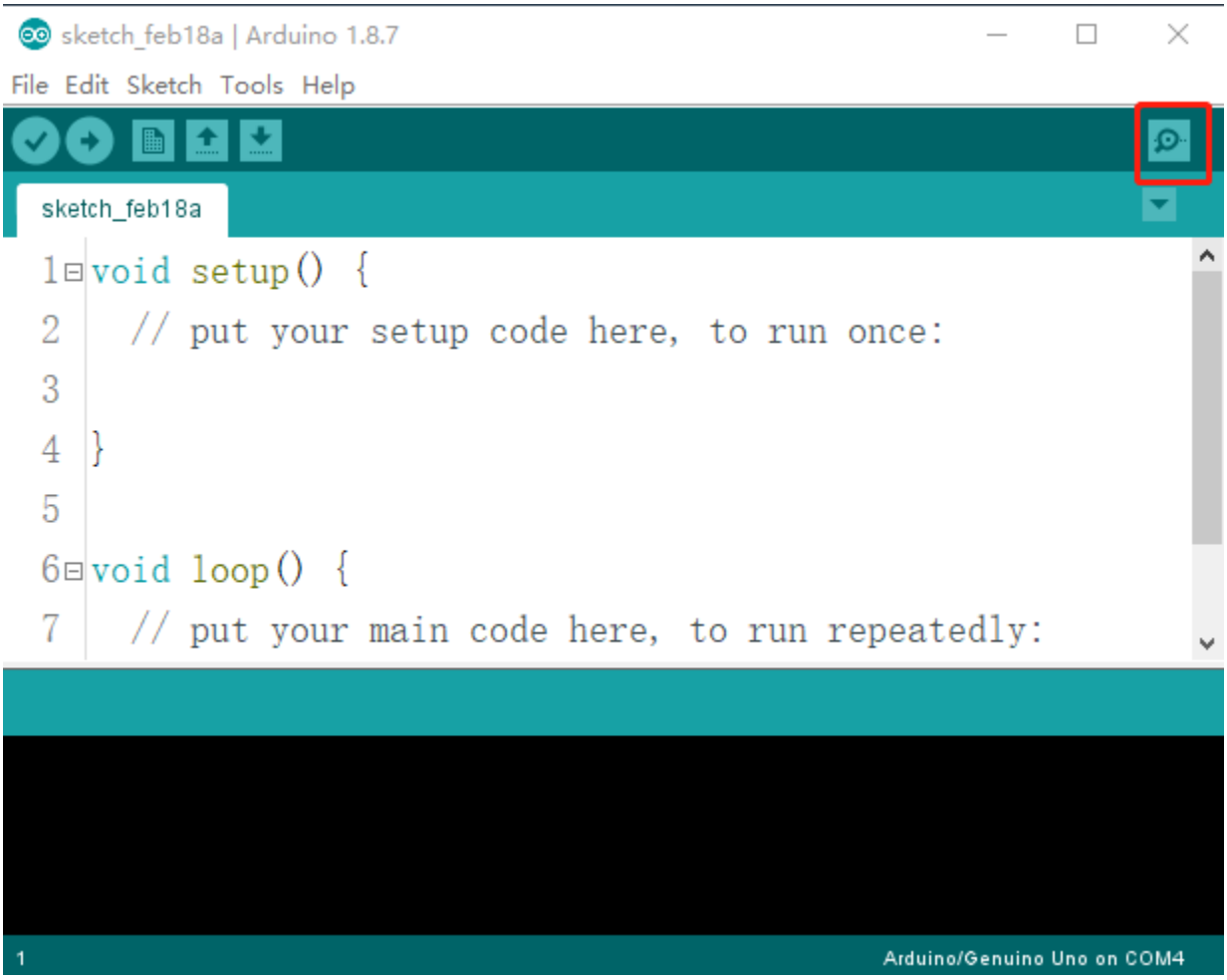
6.8.2 所需器件



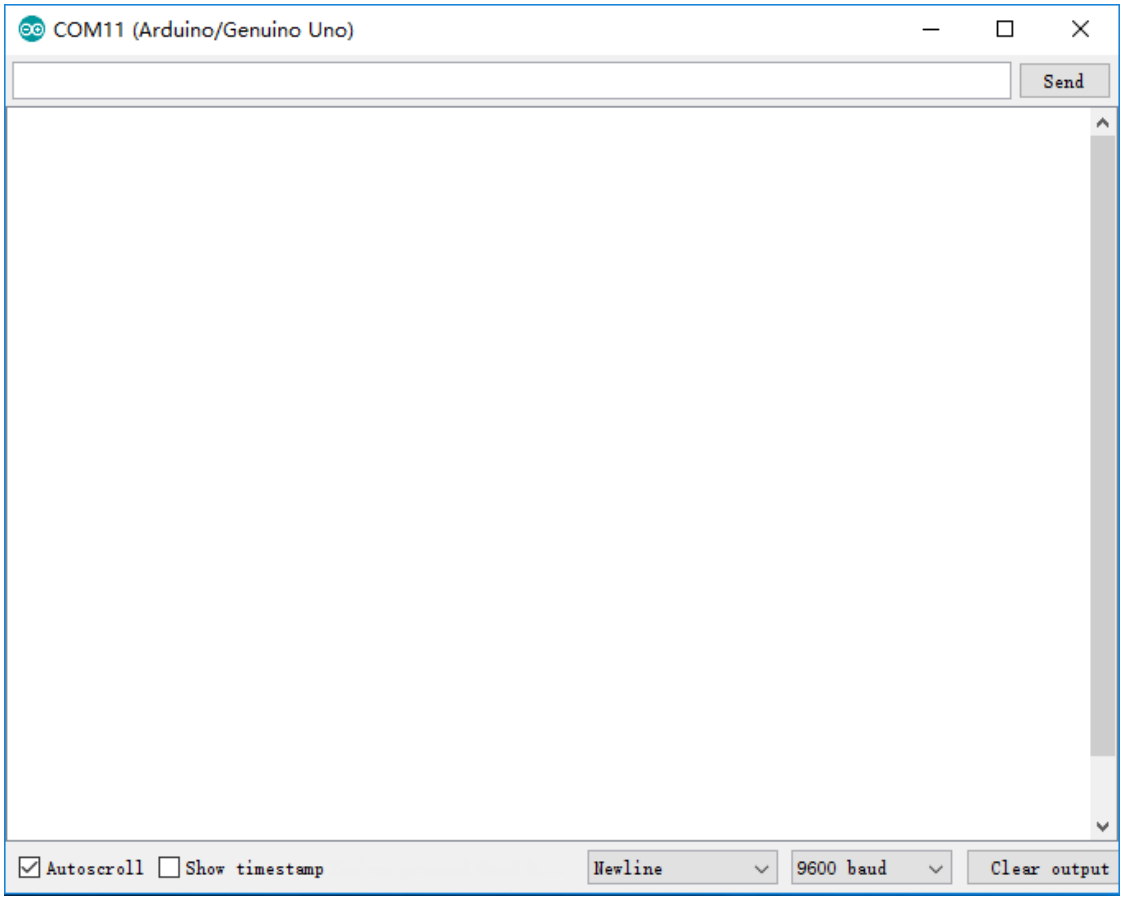
- *SunFounder R3 板*
- 面包板
- 跳线
- *LED 发光二极管*
- 电阻
- 电位器

6.8.3 串行监视器

串行监视器用于控制板和计算机或其他设备之间的通信，可以用来发送和接收数据。它是 Arduino 环境中的内置软件，你可以点击右上角的按钮打开它。

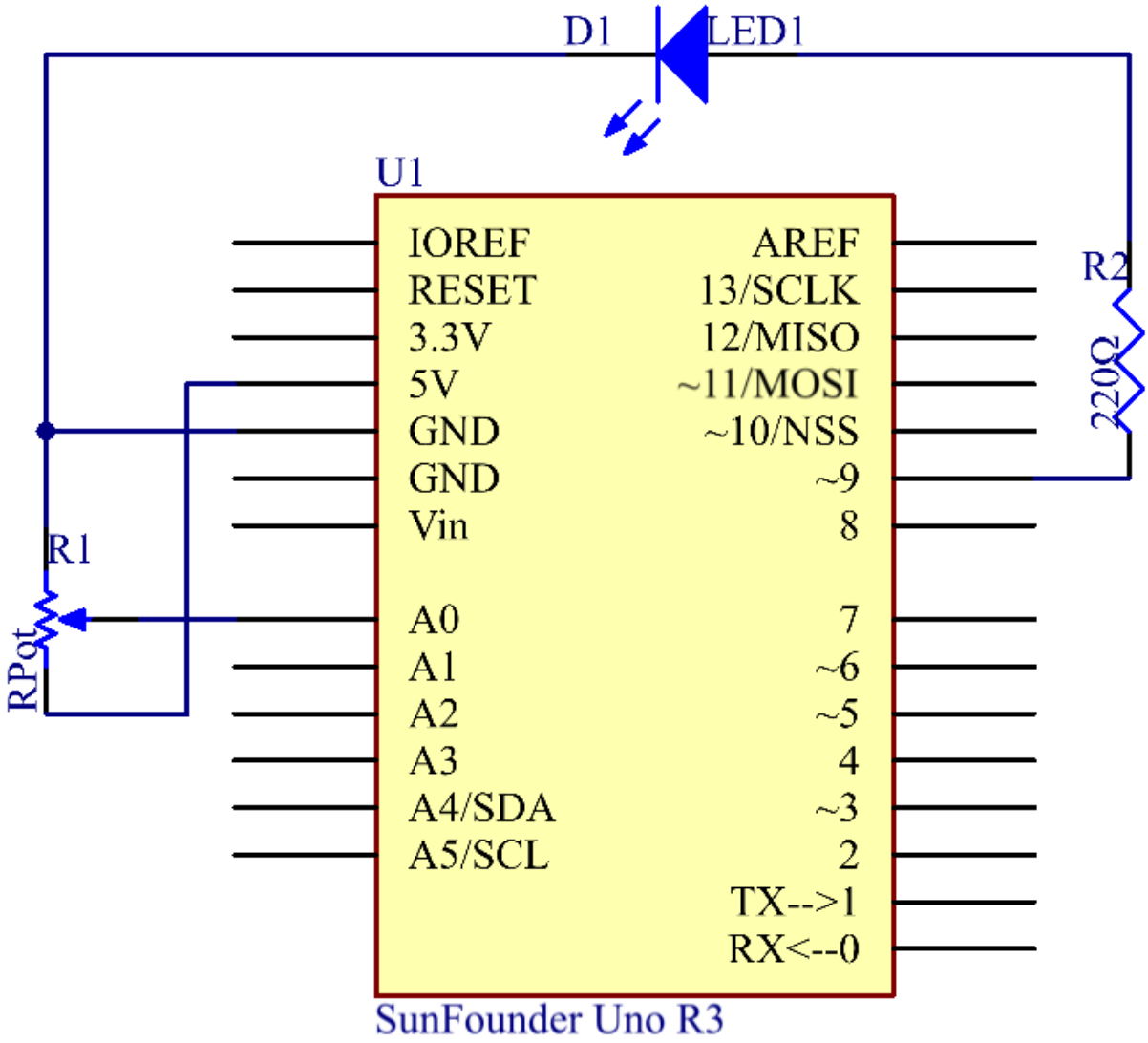


在这里，串行监视器用作计算机和控制板之间通信的中转站。首先，计算机将数据传输到 **串口监视器**，然后由控制板读取数据。最后在控制板进行相关操作。点击右上角的图标，会弹出一个窗口，如下图：



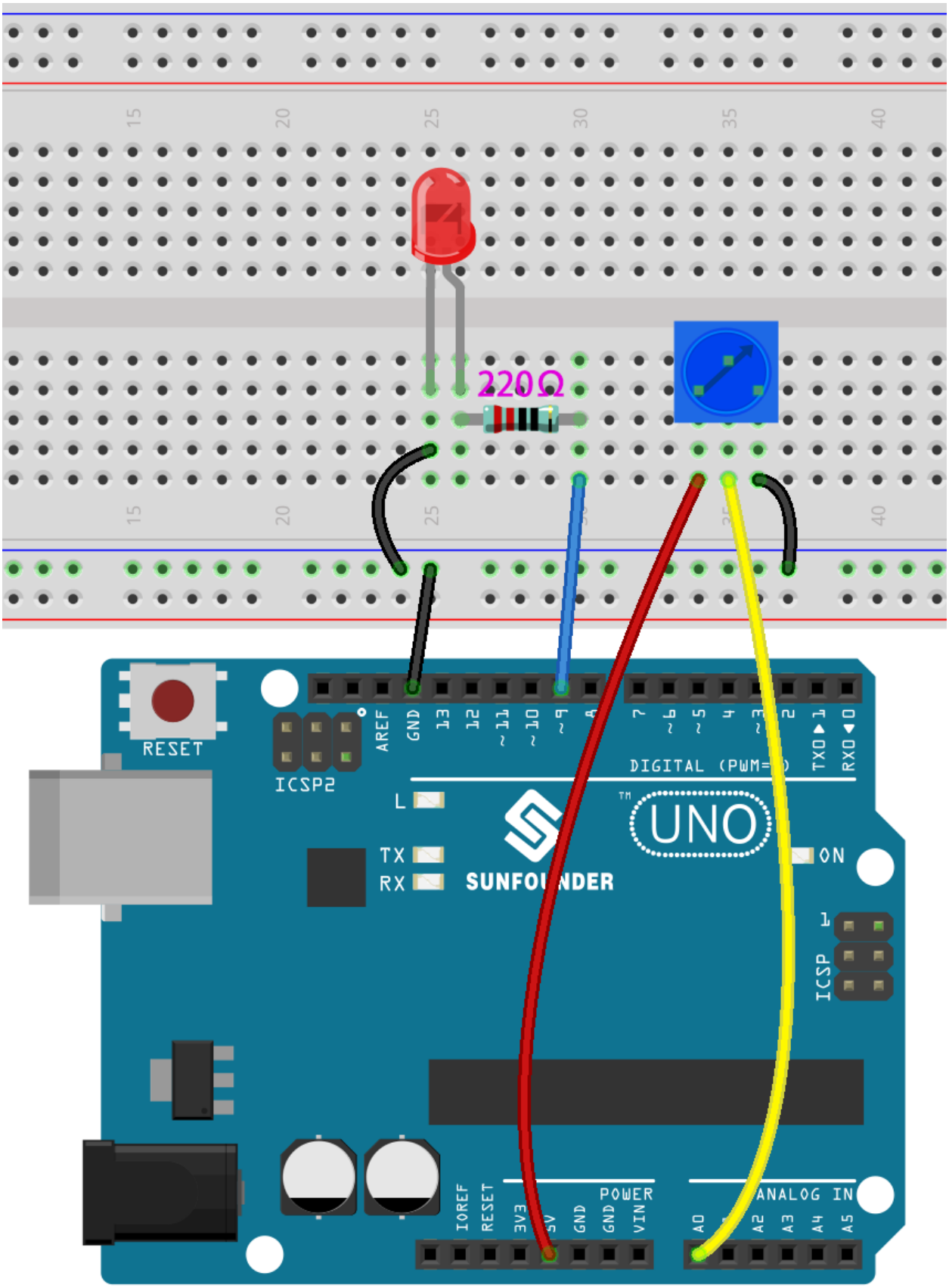
6.8.4 原理图

在这个实验中，电位器用作分压器，这意味着将设备连接到它的所有三个引脚。将电位器的中间引脚连接到引脚 A0，另外两个引脚分别连接到 5V 和 GND。因此，电位器的电压为 0-5V。旋转电位器的旋钮，A0 脚的电压会发生变化。然后使用控制板中的 AD 转换器将该电压转换为数字值 (0-1024)。通过编程，我们可以使用转换后的数字值来控制控制板上 LED 的亮度。



6.8.5 实验步骤

第 1 步：搭建电路。.


220Ω
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5

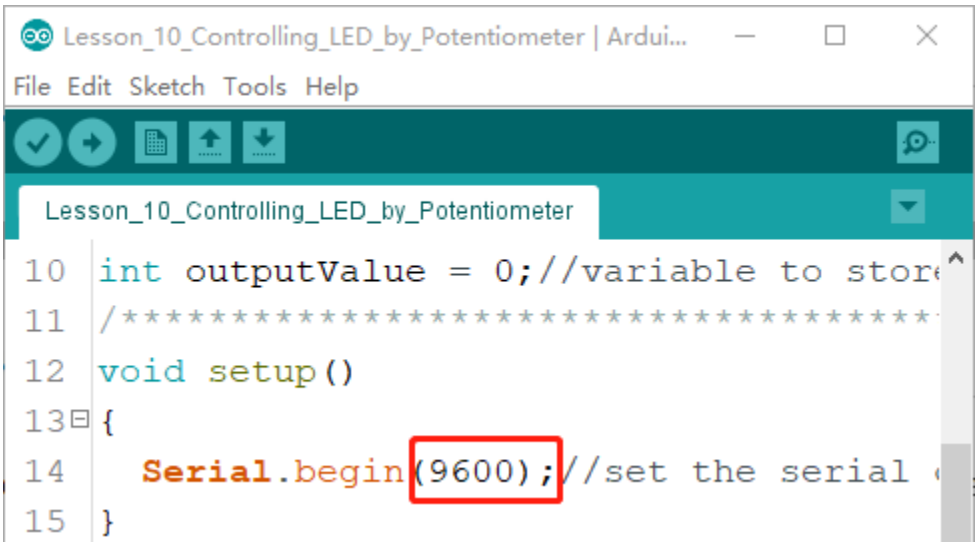
第 2 步：打开代码文件 `Lesson_8_Potentiometer.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

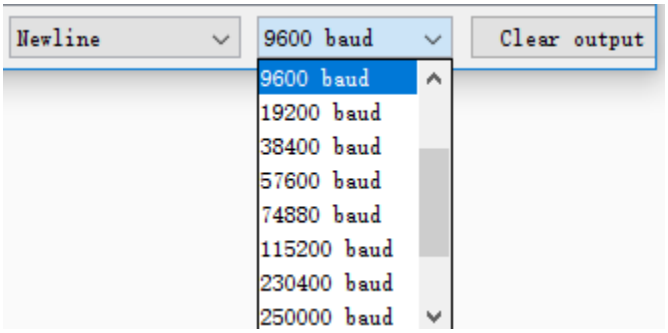
第 5 步：打开串口监视器。

找到 `Serial.begin()`，看看设置了什么波特率，这里是 9600。然后点击右上角的图标打开串口监视器。



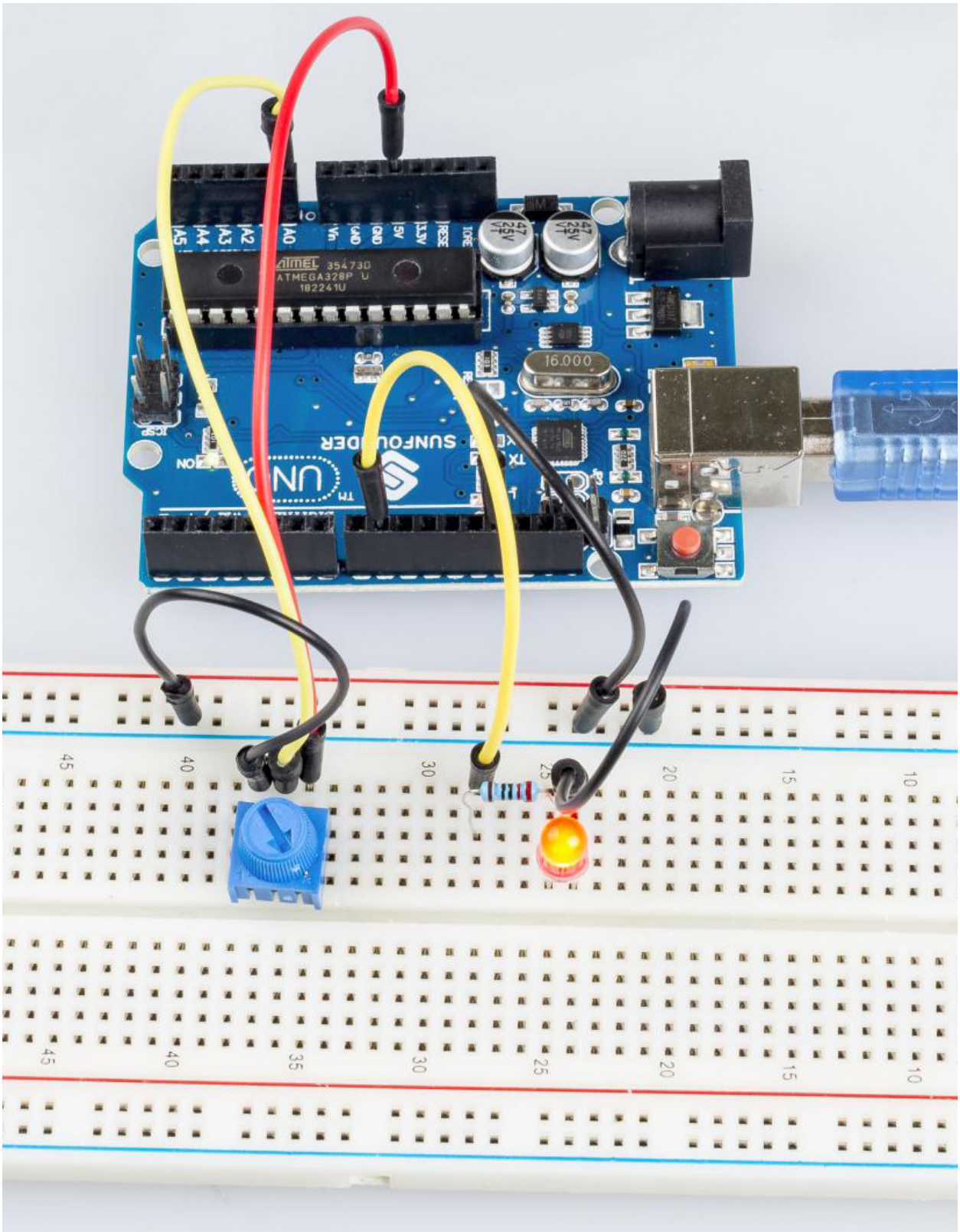
第 6 步：设置波特率为 9600。

串口监视器的默认波特率为 9600，如果代码也设置为 9600，则无需更改波特率栏。



旋转电位器的轴，你应该看到 LED 的亮度发生变化。

如果要检查相应值的变化，请打开串行监视器，窗口中的数据将随着你旋转电位器旋钮而变化。


ATMEGA328P
16.000
SUNFOUNDER
UNO

6.8.6 代码

6.8.7 代码分析

从 A0 读取值

```
inputValue = analogRead(analogPin);//read the value from the potentiometer
```

这一行是将 A0 读取的值存储在之前定义的 `inputValue` 中。

`analogRead()` 从指定的模拟引脚读取值。这意味着它会将 0 到 5 伏之间的输入电压映射为 0 到 1023 之间的整数值。

在串行监视器上打印值

```
Serial.print("Input: "); //print "Input"
Serial.println(inputValue); //print inputValue
```

- `Serial.print()`：将数据作为人类可读的 ASCII 文本打印到串口。这个命令可以有多种形式。数字被打印为每个数字的 ASCII 字符。浮点数同样被打印为 ASCII 数字，默认为两位小数。字节以单个字符的形式发送。字符和字符串按原样发送。
- `Serial.println()`：与 `Serial.print()` 相同，但它后面有一个回车字符（ASCII 13，或'r'）和一个换行字符（ASCII 10，或'n'）。

将值映射

```
outputValue = map(inputValue, 0, 1023, 0, 255); //Convert from 0-1023 proportional to␣
↪the number of a number of from 0 to 255
```

- `map(value, fromLow, fromHigh, toLow, toHigh)` 函数是将数字从一个范围重新映射到另一个范围。也就是说，值 fromLow 将被映射到了 tolow，值 fromHigh 到 toHigh，值之间以值之间，等等。

由于 `ledPin` 的范围是 0-255，我们需要将 0-1023 映射到 0-255。

以同样的方式在串口监视器中显示输出值。如果你对 `map()` 函数不是很清楚，你可以观察串口监视器中的数据并进行分析。

```
Serial.print("Output: "); //print "Output"

Serial.println(outputValue); //print outputValue
```

将电位器的值写到 LED 上

```
analogWrite(ledPin, outputValue); //turn the LED on depending on the output value
```

将输出值写入 `ledPin`，你将看到 LED 的亮度随着你旋转电位器旋钮而变化。

- `analogWrite()`：将模拟值（PWM 波）写入引脚。它与模拟引脚无关，仅适用于 PWM 引脚。在调用 `analog Write()` 之前，你不需要调用 `pinMode()` 将引脚设置为输出。

6.8.8 实验总结

这个实验也可以随意改成其他的。例如，使用电位器来控制 LED 闪烁的时间间隔。就是利用从电位器读取的数值进行延时，如下图。试试!

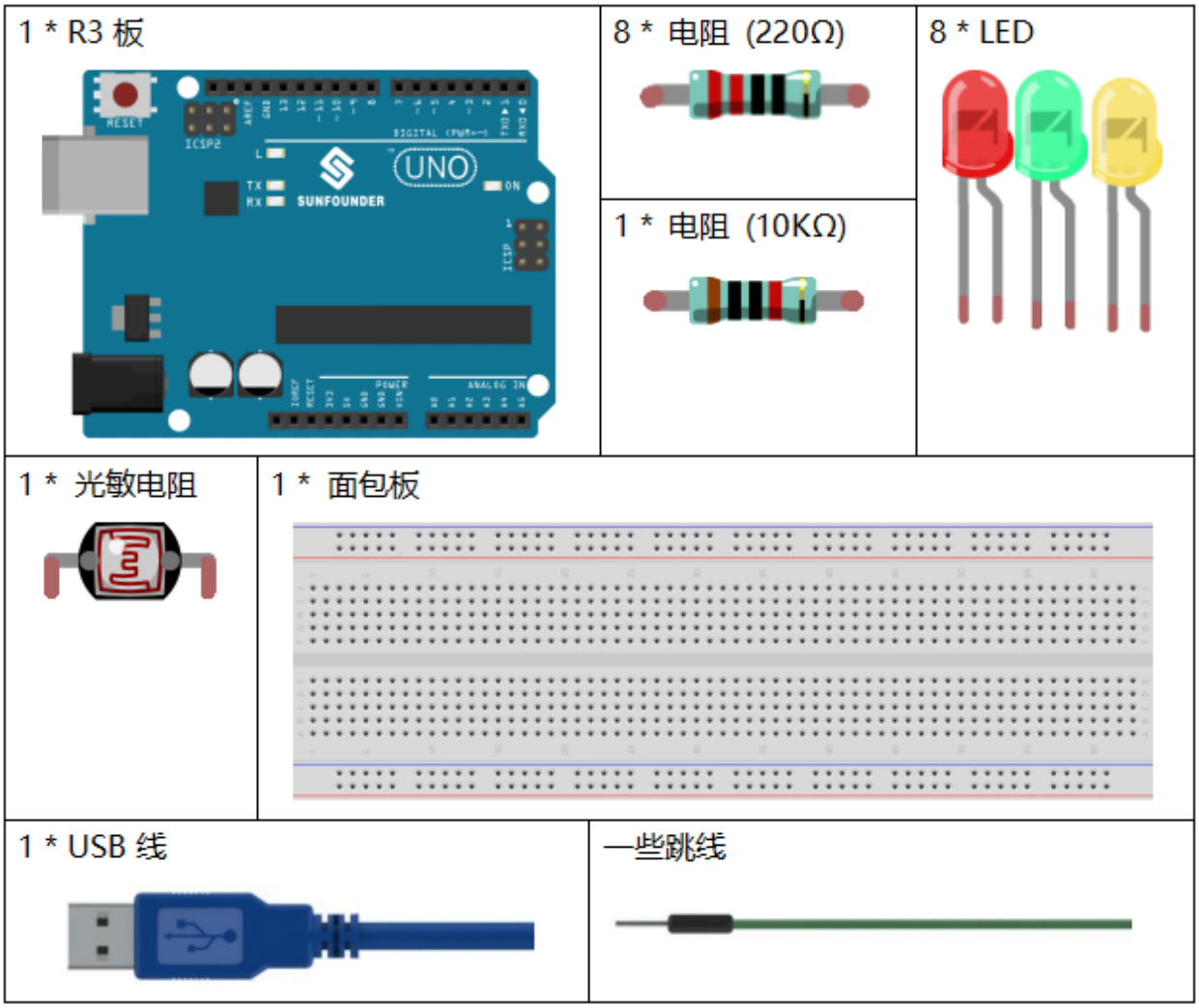
```
inputValue = analogRead(analogPin);
digitalWrite(ledPin, HIGH);
delay(inputValue);
digitalWrite(ledPin, LOW);
delay(inputValue);
```

6.9 第 9 课光敏电阻

6.9.1 介绍

在本课中，你将学习如何使用光敏电阻测量光强度。光敏电阻的电阻值随着入射光强度的变化而变化。如果光照强度变高，阻值减少；如果光强度变暗，阻值就会增加。

6.9.2 所需器件

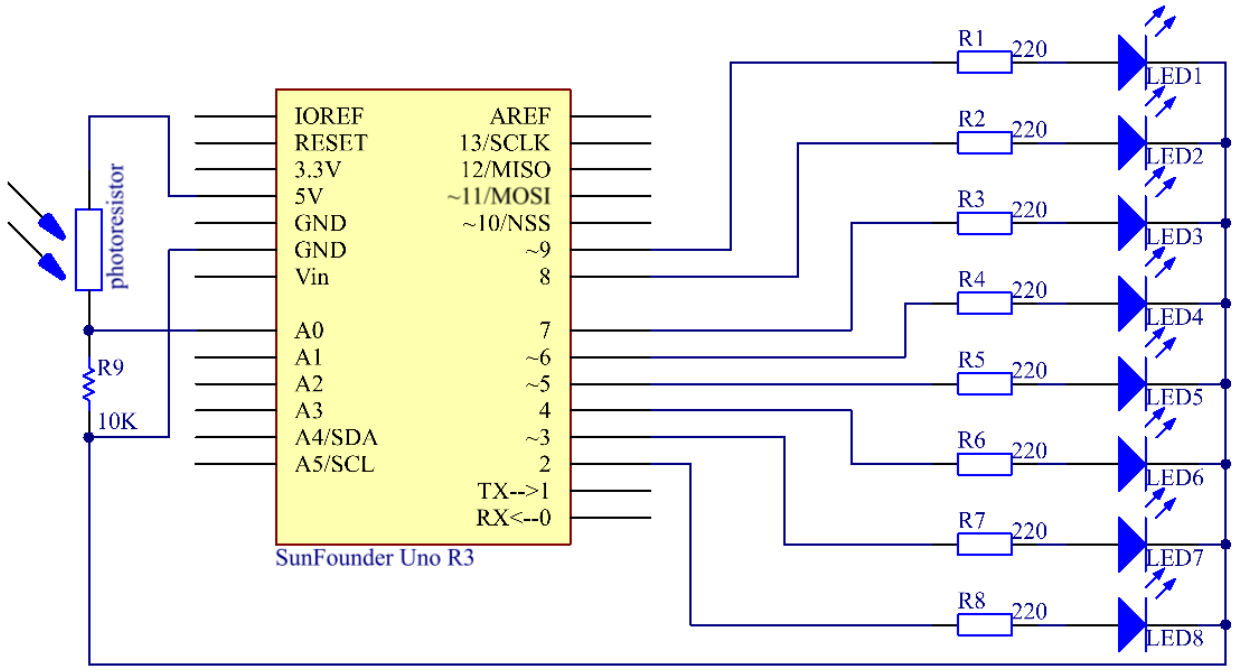


- *SunFounder R3 板*
- 面包板
- 跳线
- *LED 发光二极管*
- 电阻
- 光敏电阻

6.9.3 原理图

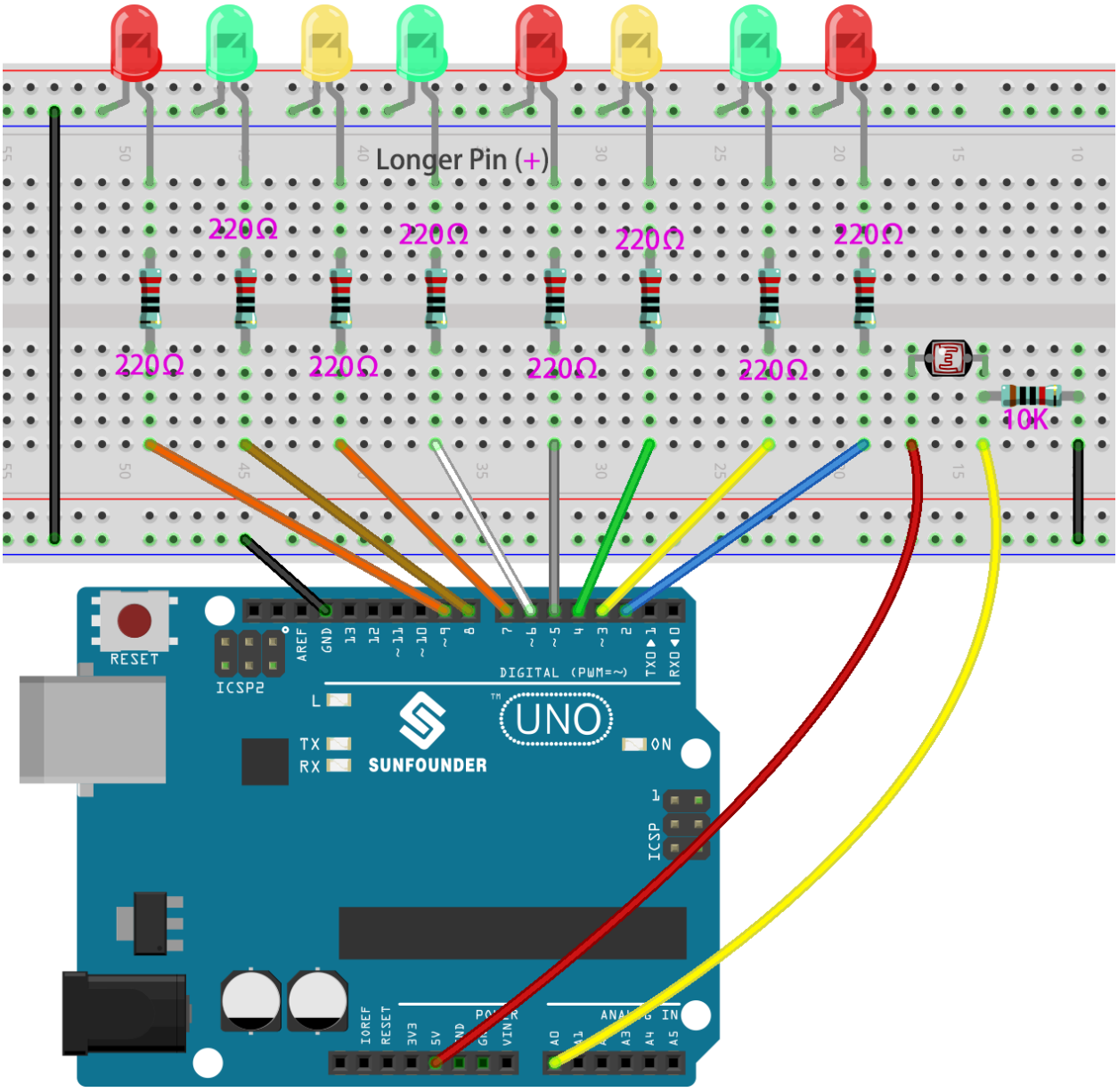
在这个实验中，我们将用 8 个 LED 灯来显示光的强度。光强度越高，越多的 LED 灯会亮起来。当光强度足够高时，所有的 LED 灯都会亮起来。当没有光，所有的 LED 灯都会熄灭。

原理图如下所示：



6.9.4 实验步骤

第 1 步：搭建电路。.

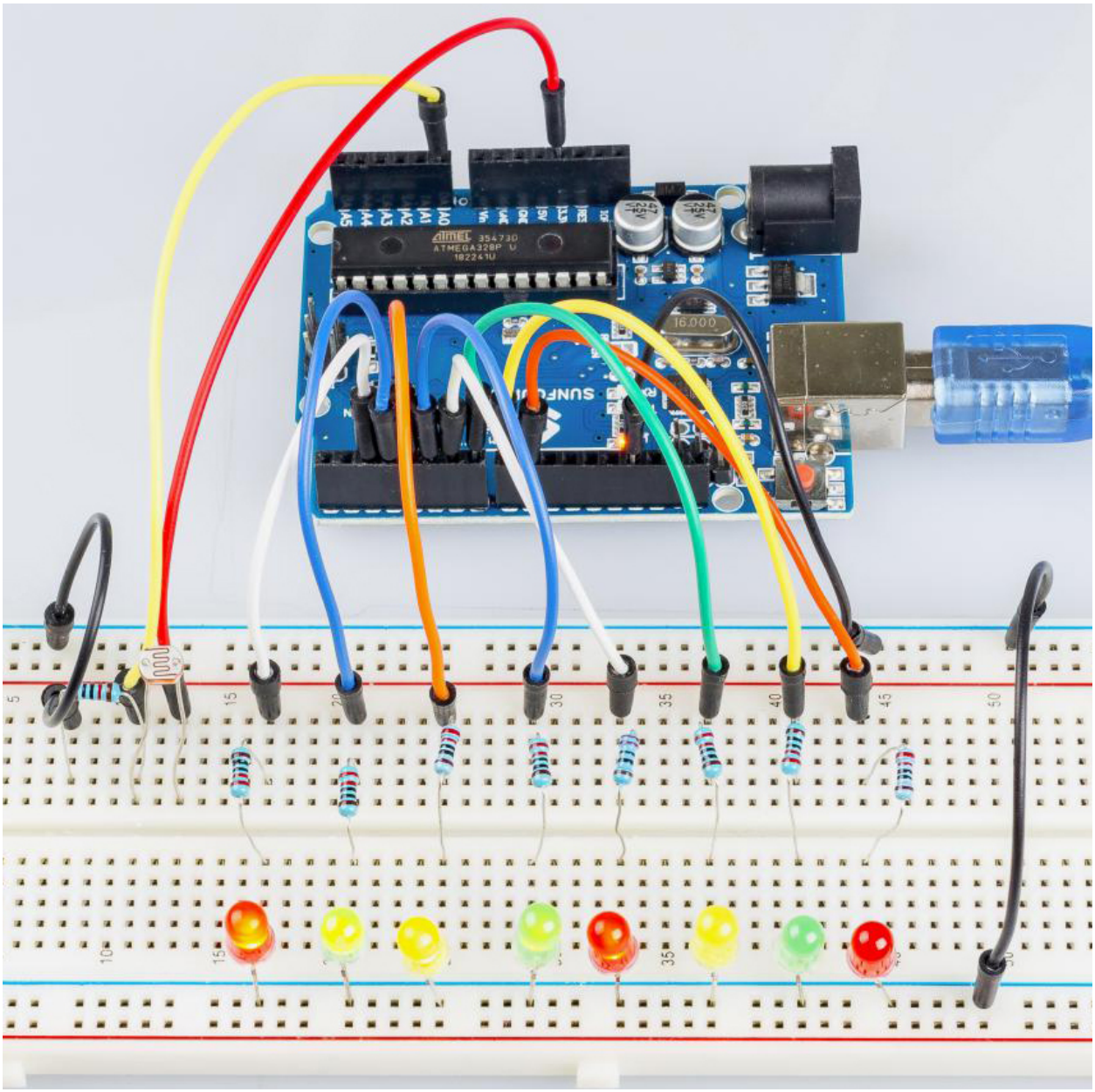


第 2 步：打开代码文件 `Lesson_9_Photoresistor.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，用一些光照在光敏电阻上，你会看到几个 LED 灯亮起来。照更多的光，你会看到更多的 LED 灯亮起来。当你把它放在一个黑暗的环境中，所有的 LED 都会熄灭。



6.9.5 代码

6.9.6 代码分析

设置变量

```
const int NbrLEDs = 8; // 8 leds
const int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9}; // 8 leds attach to pin 5-12␣
↪respectively
const int photocellPin = A0; // photoresistor attach to A0
int sensorValue = 0; // value read from the sensor
int ledLevel = 0; // sensor value converted into LED 'bars'
```

8 个 LED 被连接到 5 引脚-12 引脚，在这段代码中，使用一个数组来存储这些引脚，`ledPins[0]` 等于 5，`ledPins[1]` 等于 6，以此类推。

设置 8 个引脚为输出

```
for (int led = 0; led < NbrLEDs; led++)
{
    pinMode(ledPins[led], OUTPUT); // make all the LED pins outputs
}
```

使用 `for()` 语句将 8 个引脚依次设置为输出。依次为 OUTPUT。

读取光敏电阻的模拟值。

```
sensorValue = analogRead(photocellPin); // read the value of A0
```

读取 `photocellPin` (A0 引脚) 的值并存储到变量 `sensorValue` 中。

- `analogRead()`：从指定的模拟引脚读取数值。Arduino 板包含一个多通道、10 位的模拟数字转换器。这意味着它将映射出 0 到工作电压（5V 或 3.3V）之间的输入电压。电压（5V 或 3.3V）之间的输入电压映射为 0 至 1023 之间的整数值。

```
Serial.print("SensorValue: ");
Serial.println(sensorValue); // Print the analog value of the photoresistor
```

使用 `Serial.print()` 函数来打印光敏电阻的模拟值，你将在串口监视器中看到它们。

- `Serial.print()`：将数据作为人类可读的 ASCII 文本打印到串口。这个命令可以有多种形式。数字被打印为每个数字的 ASCII 字符。浮点数同样被打印为 ASCII 数字，默认为两位小数。字节以单个字符的形式发送。字符和字符串按原样发送。
- `Serial.println()`：与 `Serial.print()` 相同，但它后面有一个回车字符（ASCII 13，或'r'）和一个换行字符（ASCII 10，或'n'）。

将模拟值映射到 8 个 LED 上

```
ledLevel = map(sensorValue, 0, 1023, 0, NbrLEDs); // map to the number of LEDs
Serial.print("ledLevel: ");
Serial.println(ledLevel);
```

这个 `map()` 函数是用来将 0-1023 映射到 0-NbrLEDs(8)。

(1023-0)/(8-0)=127.875

0-12	128-2	2 56-38	384-	5 12-63	640-7	7 68-89	896 -
7.875	55.75	3.625	511.5	9.375	67.25	5.125	1023
0	1	2	3	4	5	6	7

如果 `sensorValue` 等于 560，则 `ledLevel` 为 4.

- `map(value, fromLow, fromHigh, toLow, toHigh)` 函数是将数字从一个范围重新映射到另一个范围。也就是说，值 fromLow 将被映射到了 tolow，值 fromHigh 到 toHigh，值之间以值之间，等等。

点亮 LED 灯

```
for (int led = 0; led < NbrLEDs; led++)
{
    if (led <= ledLevel ) //When led is smaller than ledLevel, run the following code.
    {
```

(续下页)

(接上页)

```
        digitalWrite(ledPins[led], HIGH); // turn on pins less than the level
    }
    else
    {
        digitalWrite(ledPins[led], LOW); // turn off pins higher than
    }
}
```

点亮相应的 LED。例如，当 `ledLevel` 为 4 时，点亮 `ledPins[0]` 到 `ledPins[4]`，熄灭 `ledPins[5]` 到 `ledPins[7]`。

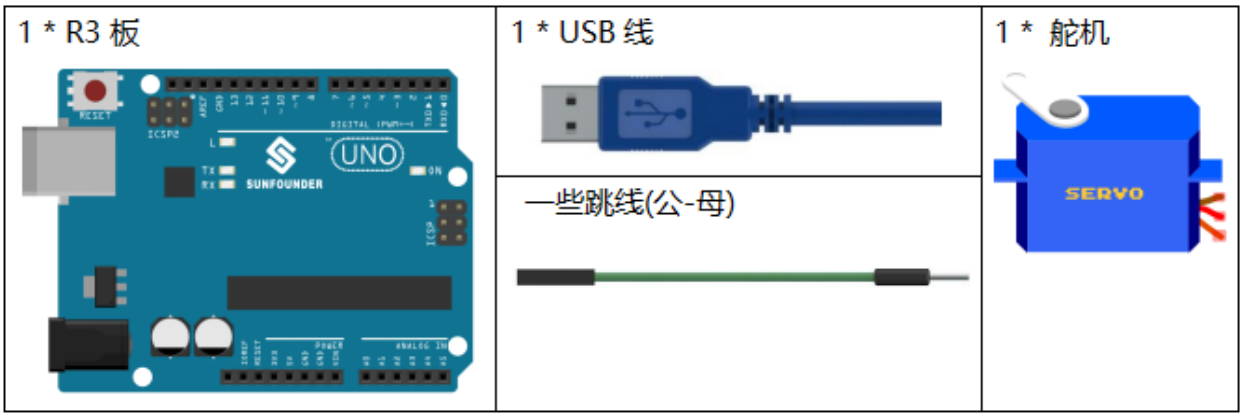
6.10 第 10 课舵机

6.10.1 介绍

伺服是一种只能旋转 180 度的减速电机。它是通过从你的电路板发送电脉冲来控制的。这些脉冲告诉伺服它应该移动到什么位置。

舵机有三根线：棕色线为 GND，红色线为 VCC，橙色线为信号线。

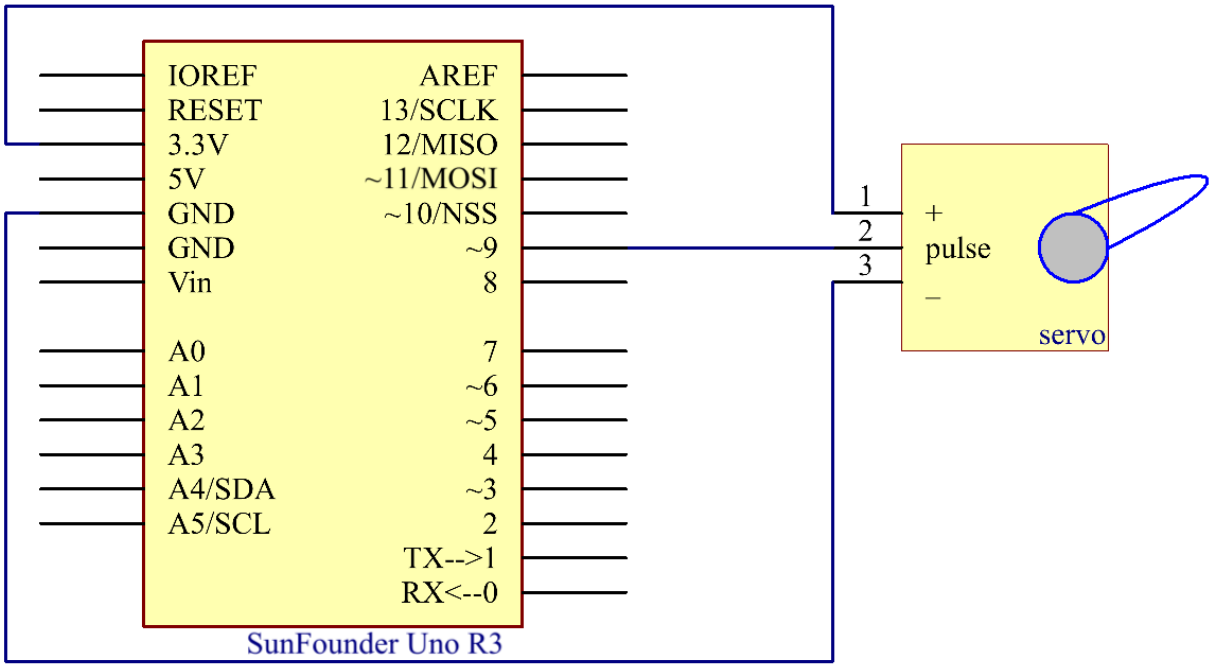
6.10.2 所需器件



- *SunFounder R3 板*
- *面包板*
- *跳线*
- *舵机*

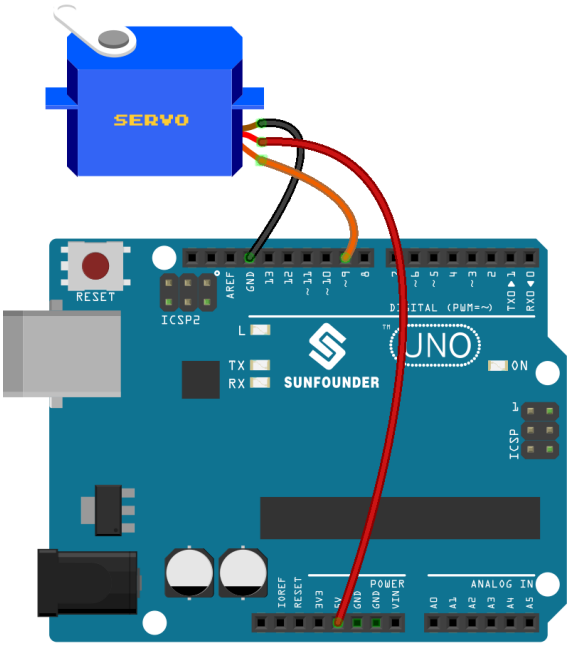
6.10.3 原理图

原理图如下所示：



6.10.4 实验步骤

第 1 步：搭建电路。(棕色连接到 GND，橙色连接到 9 引脚，红色连接到 5V)。

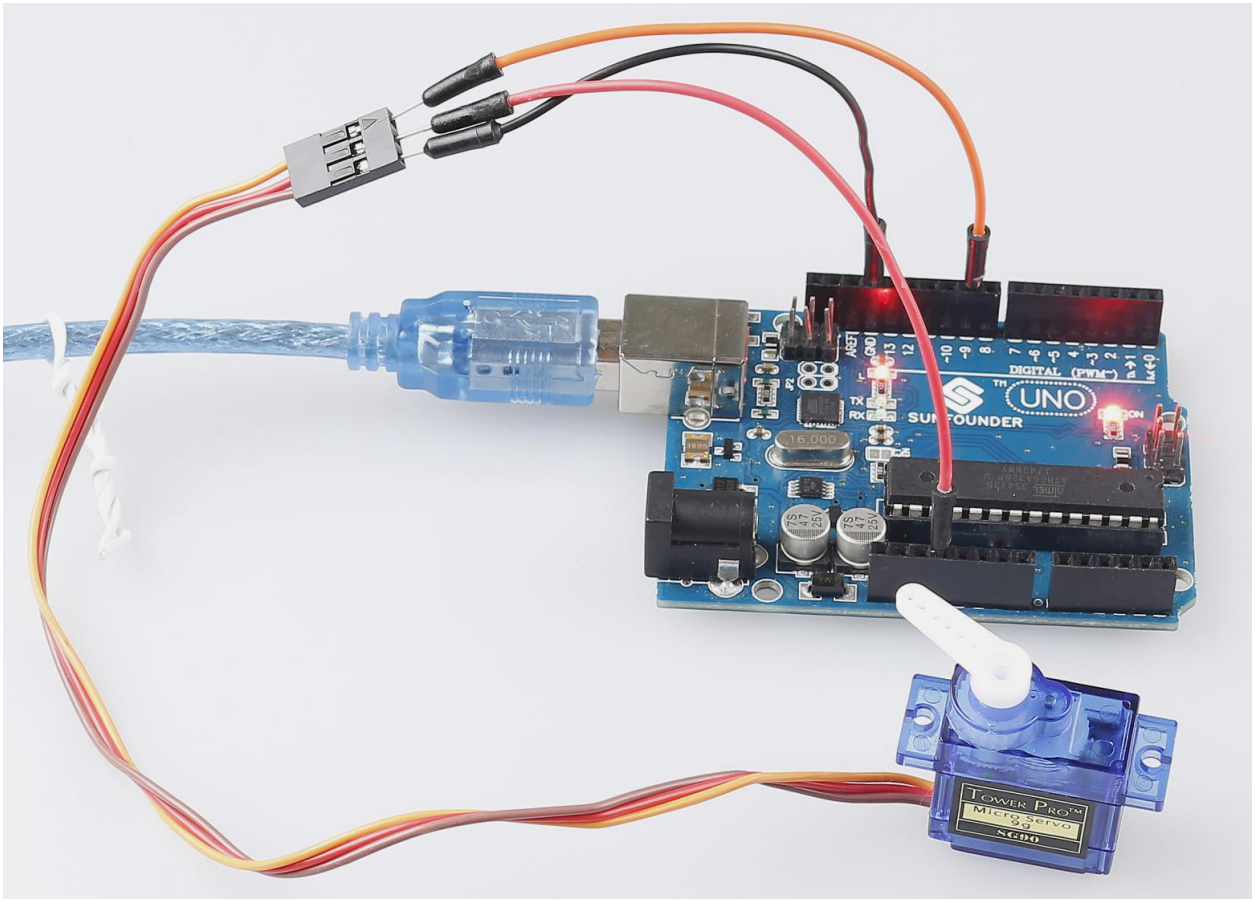


第 2 步：打开代码文件 `Lesson_10_Servo.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你可以看到舵机的摇臂旋转并停止在 90 度（每次 15 度）。然后它向相反的方向旋转。



6.10.5 代码

6.10.6 代码分析

添加一个库

```
#include <Servo.h>
Servo myservo; //create servo object to control a servo
```

导入 Servo.h 文件之后，你就可以在这个文件中的函数。Servo 是 Arduino IDE 中的内置库。你可以在安装路径，默认是 C:Program FilesArduinolibraries 下找到 Servo 文件夹。

初始化舵机

```
void setup()
{
    myservo.attach(9); //attachs the servo on pin 9 to servo object
    myservo.write(0); //back to 0 degrees
    delay(1000); //wait for a second
}
```

- myservo.attach()：用来初始化舵机，并设置它的信号引脚。

- `myservo.write()`：将一个值写入舵机，相应地控制它的轴。在一个标准的舵机上，这将设置轴的角度(度)，将轴移到那个方向。这里让伺服机首先保持在 0 角度。

让舵机转动

```
void loop()
{
    for (int i = 0; i <= 180; i++)
    {
        myservo.write(i); //write the i angle to the servo
        delay(15); //delay 15ms
    }
    for (int i = 180; i >= 0; i--)
    {
        myservo.write(i); //write the i angle to the servo
        delay(15); //delay 15ms
    }
}
```

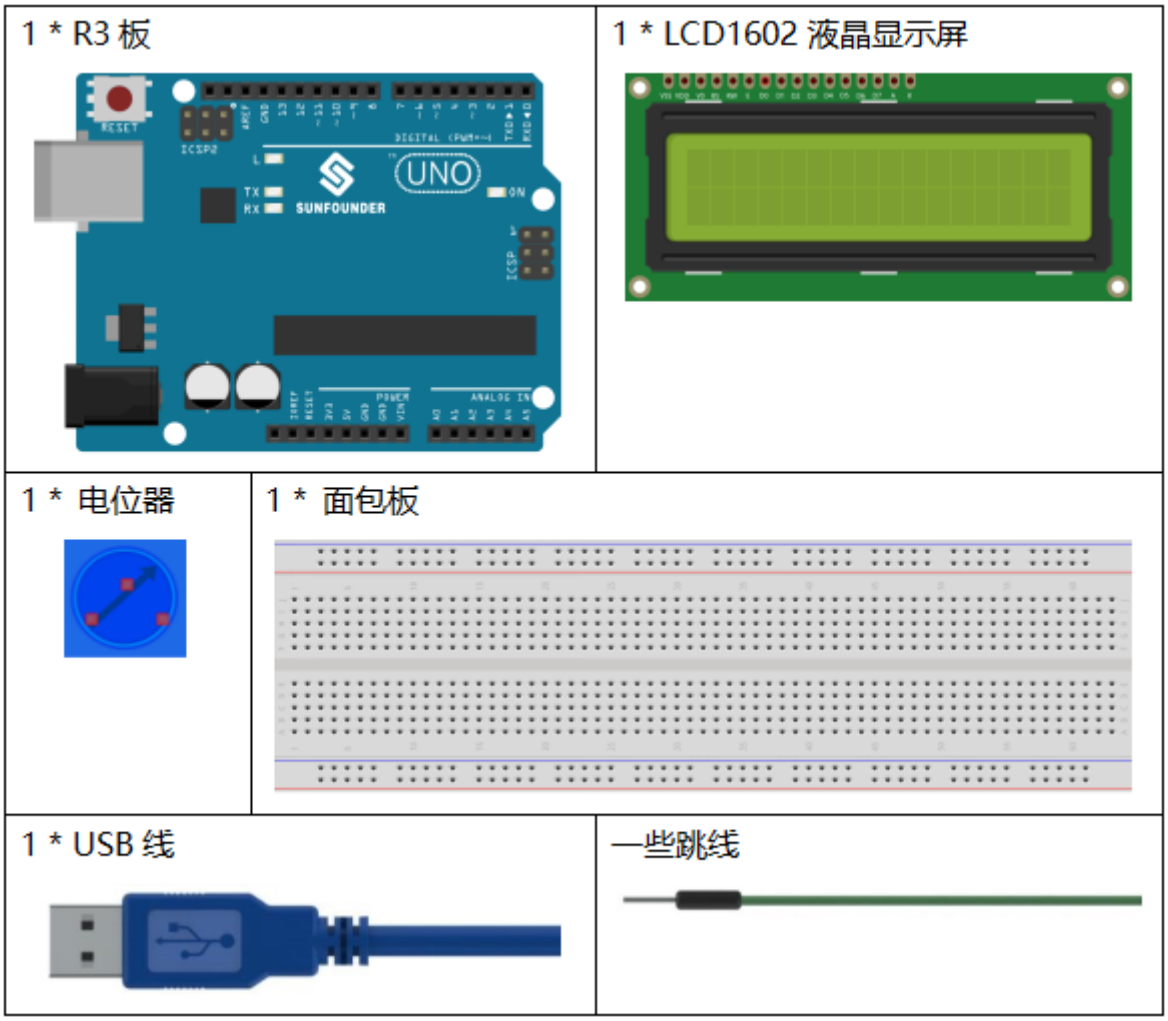
用 2 个 `for()` 语句将 0-180 写入舵机，这样就可以看到舵机从 0 转到 180 角，然后转回 0。

6.11 第 11 课 LCD1602

6.11.1 介绍

在本课中，我们将学习如何使用 LCD1602 来显示字符和字符串。LCD1602，即 1602 字符型液晶显示器，是一种显示字母、数字、字符等的点阵模块。它由 5x7 或 5x11 点阵位置组成；每个位置可以显示一个字符。两个字符之间有一个点间距，行之间有一个空格，从而将字符和行分开。数字 1602 表示在显示屏上可以显示 2 行，每行 16 个字符。现在让我们学习下如何使用它。

6.11.2 所需器件

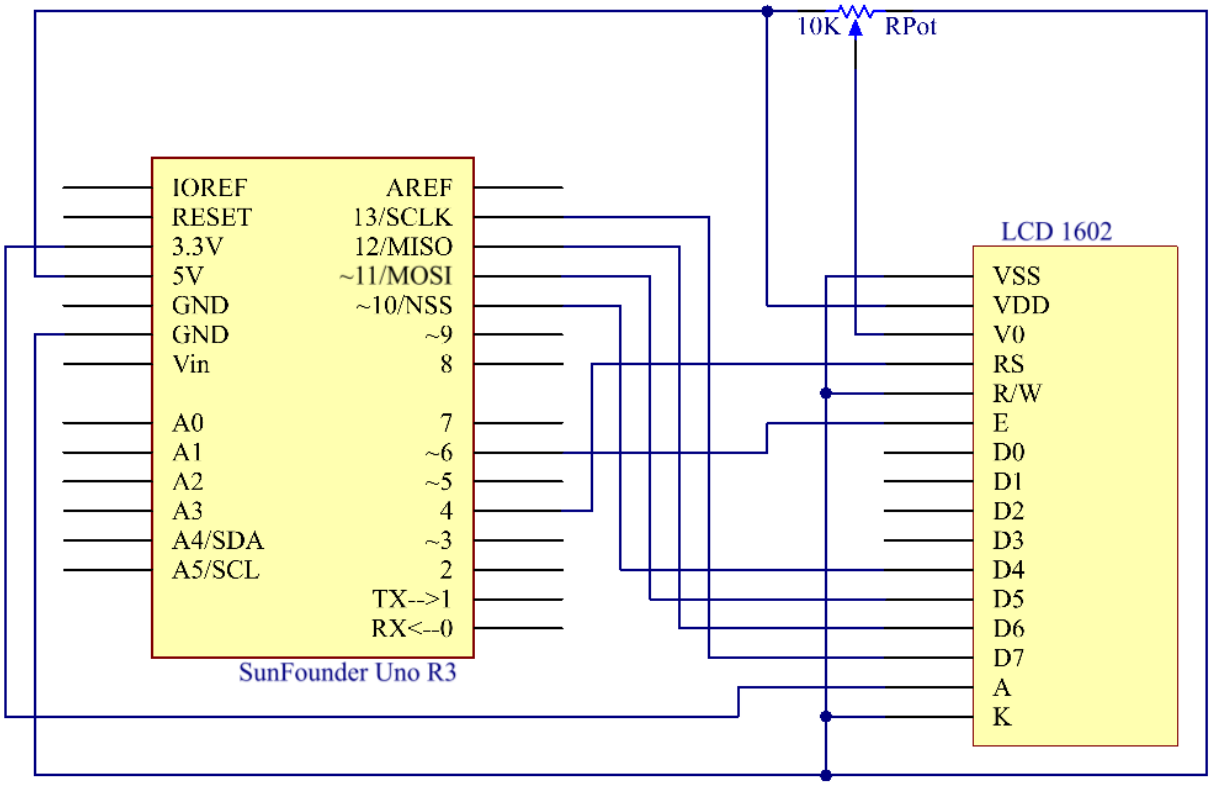


- *SunFounder R3 板*
- *面包板*
- *跳线*
- *LCD1602 液晶显示屏*
- *电位器*

6.11.3 原理图

- 连接 **K** 到 GND 和 **A** 至 3.3 V，然后将 LCD1602 的背光将被打开。
- 将 **VSS** 连接到 GND。
- 将 **VO** 连接到电位器的中间引脚 - 你可以使用它来调整屏幕显示的对比度。
- 连接 RS 到 D4 和 R / W 连接到 GND，该装置则可以写入字符的 LCD1602。
- 将 **E** 接 6 引脚，LCD1602 上显示的字符由 **D4-D7** 控制。

原理图如下所示：



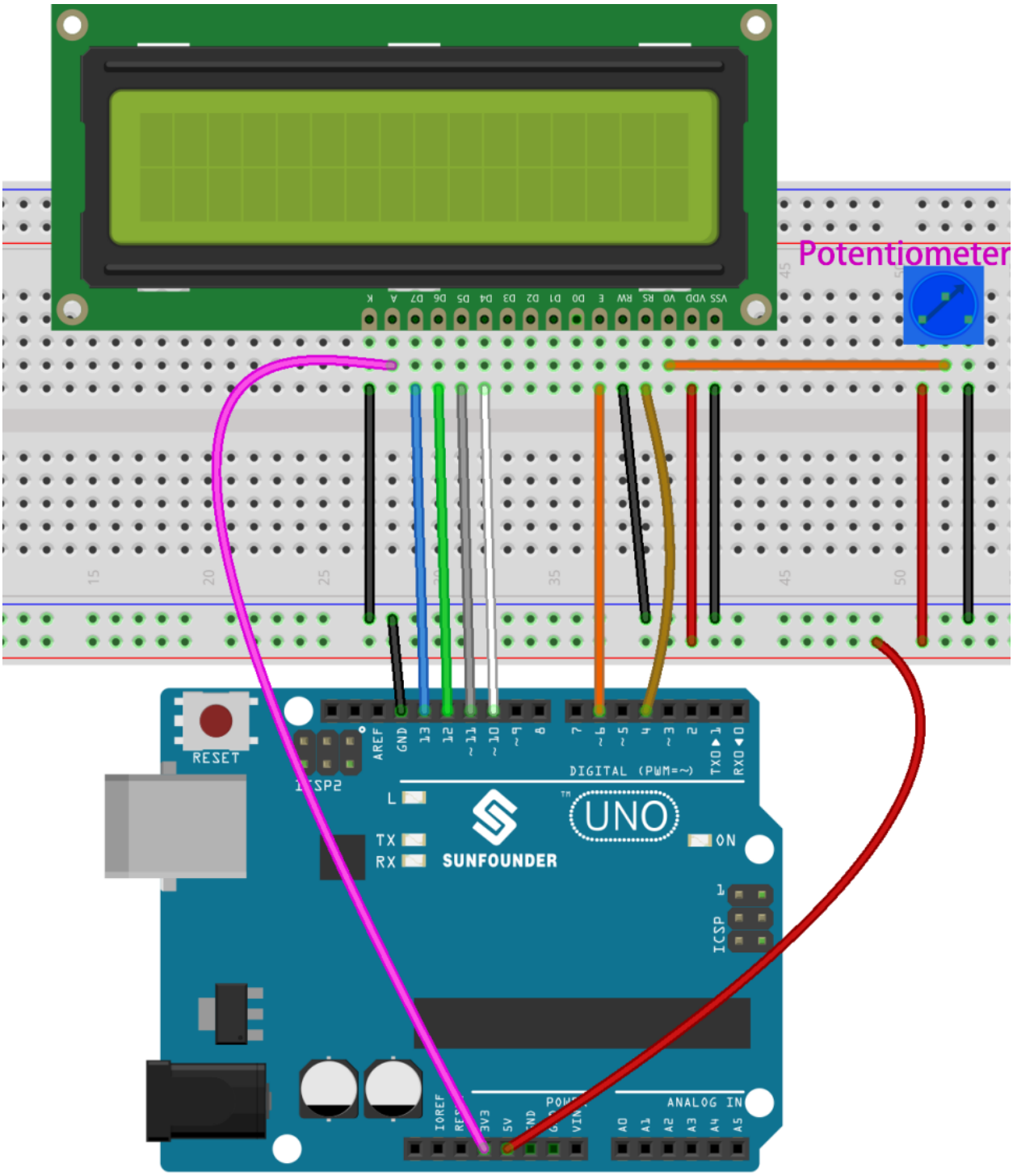
6.11.4 实验步骤

第 1 步：搭建电路。（请仔细按下图接线，以免液晶屏无法正常工作。）

第 2 步：打开代码文件 `Lesson_11_LCD1602.ino`。

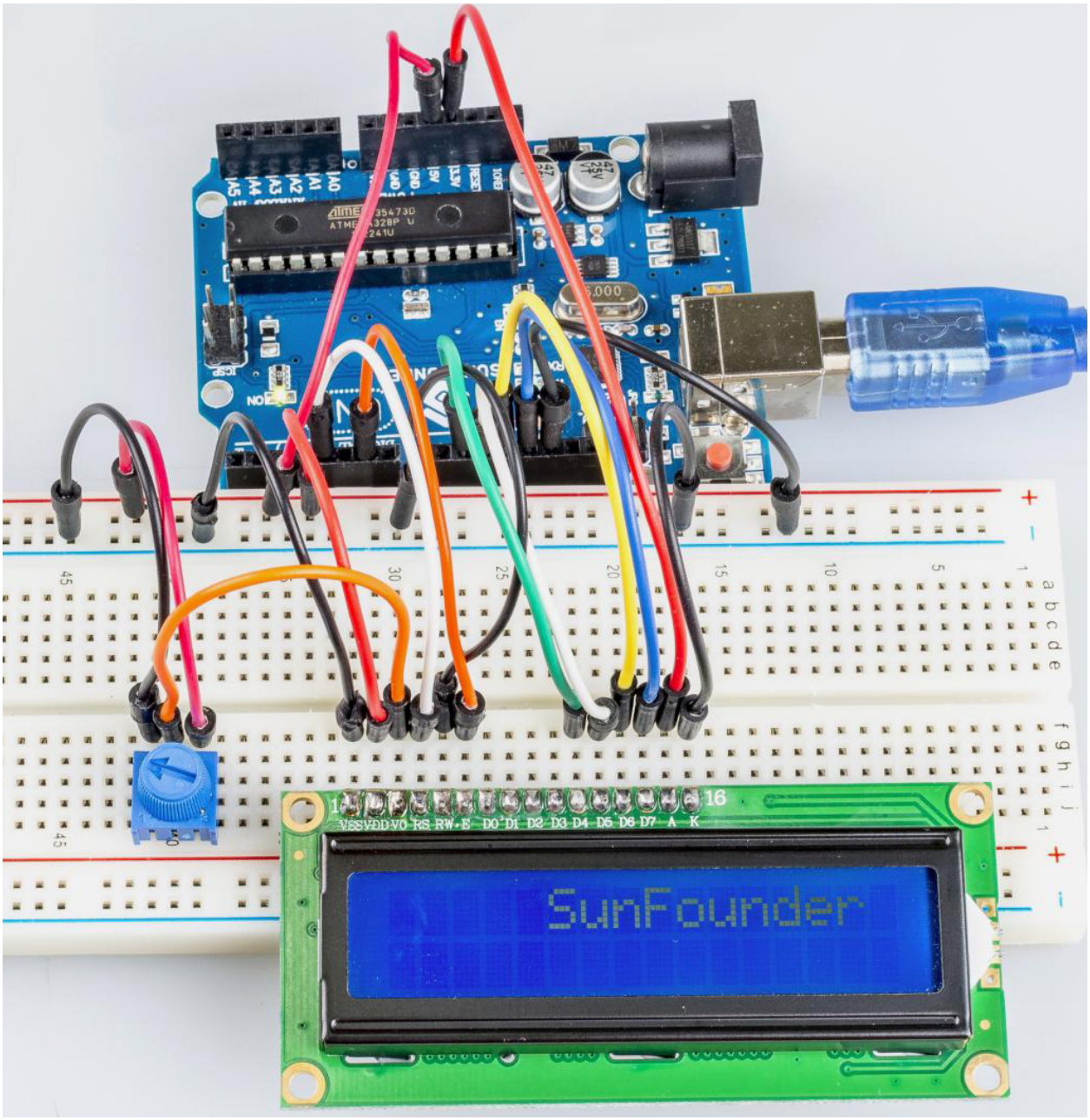
第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。



备注： 代码上传完成之后，若不显示或者显示不清晰，可旋转电位器来调整对比度（亮/暗显示比例）。

你现在应该看到字符 `SunFounder` 和 `hello, world` 在 LCD1602 上滚动。



6.11.5 代码

6.11.6 代码分析

导入一个库

```
#include <LiquidCrystal.h>// include the library code
```

由于包含了 `LiquidCrystal.h` 文件，你可以在以后调用该文件中的函数。

`LiquidCrystal` 是 Arduino IDE 中的一个内置库。你可以在安装路径，默认是 `C:\Program Files\Arduino\libraries` 下找到 `LiquidCrystal` 文件夹。

在 examples 文件夹包含的是相关的示例代码。src 文件夹包含了库的主要部分：LiquidCrystal.cpp``（执行文件，包括函数实现、变量定义等）和 ``LiquidCrystal.h``（头文件，包括函数声明、宏定义、结构定义等）。如果你想探索某个函数是如何实现的，你可以在 ``LiquidCrystal.cpp 文件中查找。

需显示的字符串

```
char array1[]=" SunFounder "; //the string to print on the LCD
char array2[]="hello, world! "; //the string to print on the LCD
```

这是两个字符型数组：arry1[] 和 array2[]。引号 "xxx " 中的内容是它们的元素，总共包括 26 个字符（空格算在内）。array1[0] 代表数组中的第一个元素，是一个空格，array1[2] 意味着第二个元素 S，以此类推。所以 array1[25] 是最后一个元素（这里也是一个空格）。

定义 LCD1602 的引脚

```
LiquidCrystal lcd(4, 6, 10, 11, 12, 13);
```

定义一个 LiquidCrystal 类型的变量 lcd。这里用 lcd 来表示下面代码中的 LiquidCrystal。

- LiquidCrysral() 函数的基本格式是：LiquidCrystal (rs, enable, d4, d5, d6, d7)。你可以查看 LiquidCrystal.cpp 文件了解详情。

所以这一行定义了 RS 脚与 4 脚相连，enable 脚与 6 脚相连，d4-d7 分别与 10-13 脚相连。

初始化 LCD1602

```
lcd.begin(16, 2); // set up the LCD's number of columns and rows: begin(col,row) is␣
↪to set the display of LCD. Here set as 16 x 2.
```

设置光标的位置

```
lcd.setCursor(15,0); // set the cursor to column 15, line 0
```

- setCursor(col,row) 用来设置光标的位置，即开始显示字符的地方。这里把它设置为 15 列（第 16 列），0 行（第 1 行）。

LCD1602 显示字符

```
for ( int positionCounter1 = 0; positionCounter1 < 26; positionCounter1++)
{
    lcd.scrollDisplayLeft(); //Scrolls the contents of the display one space to the␣
↪left.
    lcd.print(array1[positionCounter1]); // Print a message to the LCD.
    delay(tim); //wait for 250 microseconds
}
```

当 positionCounter1 = 0 时，与 positionCounter1 < 26 一致。positionCounter1 加 1，通过 lcd.scrollDisplayLeft() 向左移动一位。通过 lcd.print(array1[positionCounter1]) 使 LCD 显示 array1[0]，并延迟 tim ms（250ms）。循环 26 次后，array1[] 中的所有元素都被显示。

```
lcd.clear(); //Clears the LCD screen.
```

用 lcd.clear() 清除屏幕，这样它就不会影响下次的显示了。

```
lcd.setCursor(15,1); // set the cursor to column 15, line 1 // Set the cursor at Col.␣
↪15 Line 1, where the characters will start to show.
for (int positionCounter2 = 0; positionCounter2 < 26; positionCounter2++)
{
    lcd.scrollDisplayLeft(); //Scrolls the contents of the display one space to the␣
```

(续下页)

(接上页)

```
↪left.
    lcd.print(array2[positionCounter2]); // Print a message to the LCD.
    delay(tim); //wait for 250 microseconds
}
```

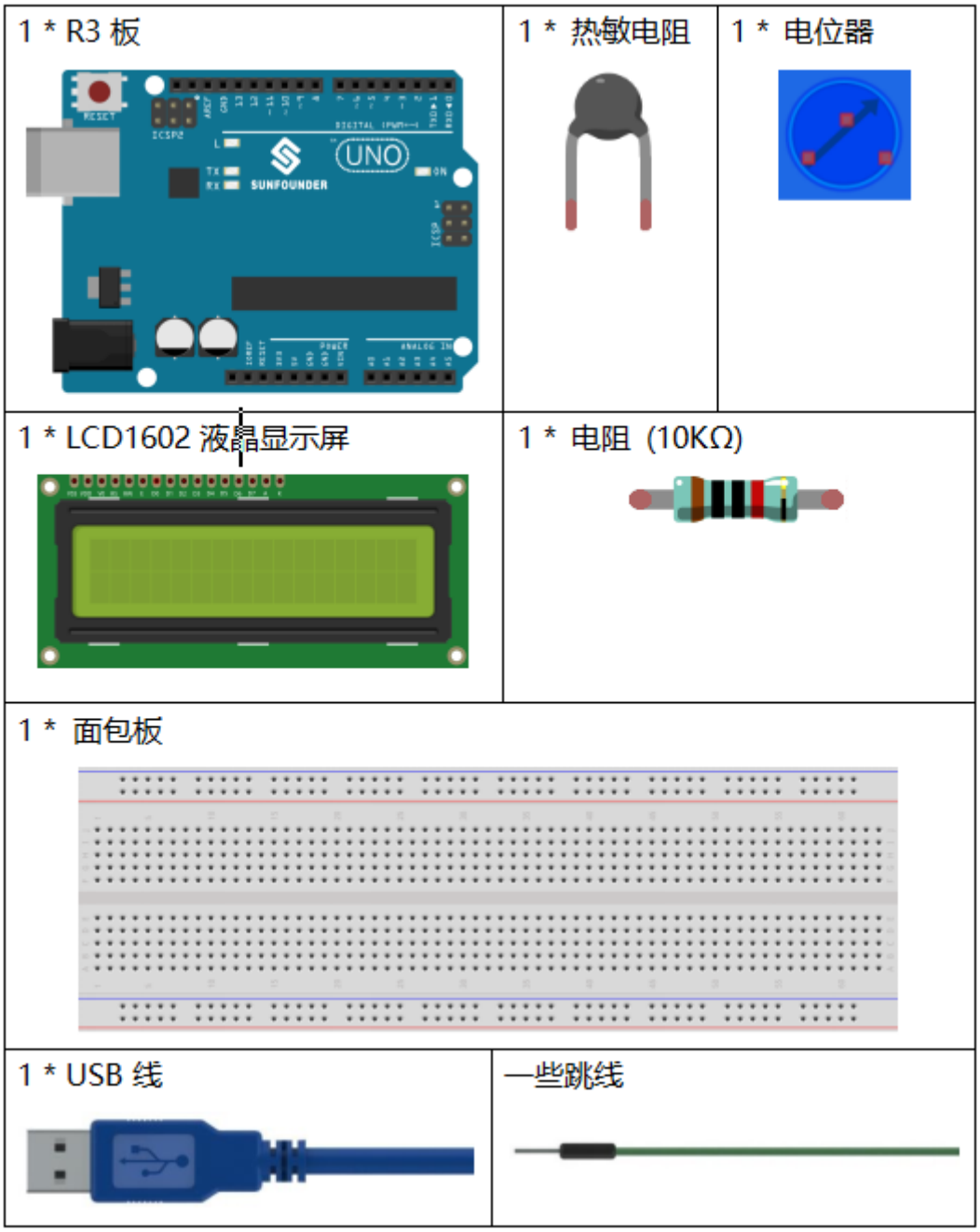
同样地，代码是在 LCD 上显示 `array2[]` 中的元素。因此，你会看到 `SunFounder` 在 LCD 的第一行向左移动直到消失。然后在第二行显示 `hello, world !`，同时也向左滚动直到消失。

6.12 第 12 课热敏电阻

6.12.1 介绍

到目前为止，我们已经学习了很多设备。要做更多的东西，你需要掌握更多的知识。今天我们要认识一个热敏电阻。它类似于光敏电阻，能够根据外部变化来改变其电阻。与光敏电阻不同，热敏电阻的电阻值随外部环境温度的变化而显着变化。

6.12.2 所需器件



- *SunFounder R3 板*
- 面包板
- 跳线
- 电阻
- 电位器
- 热敏电阻

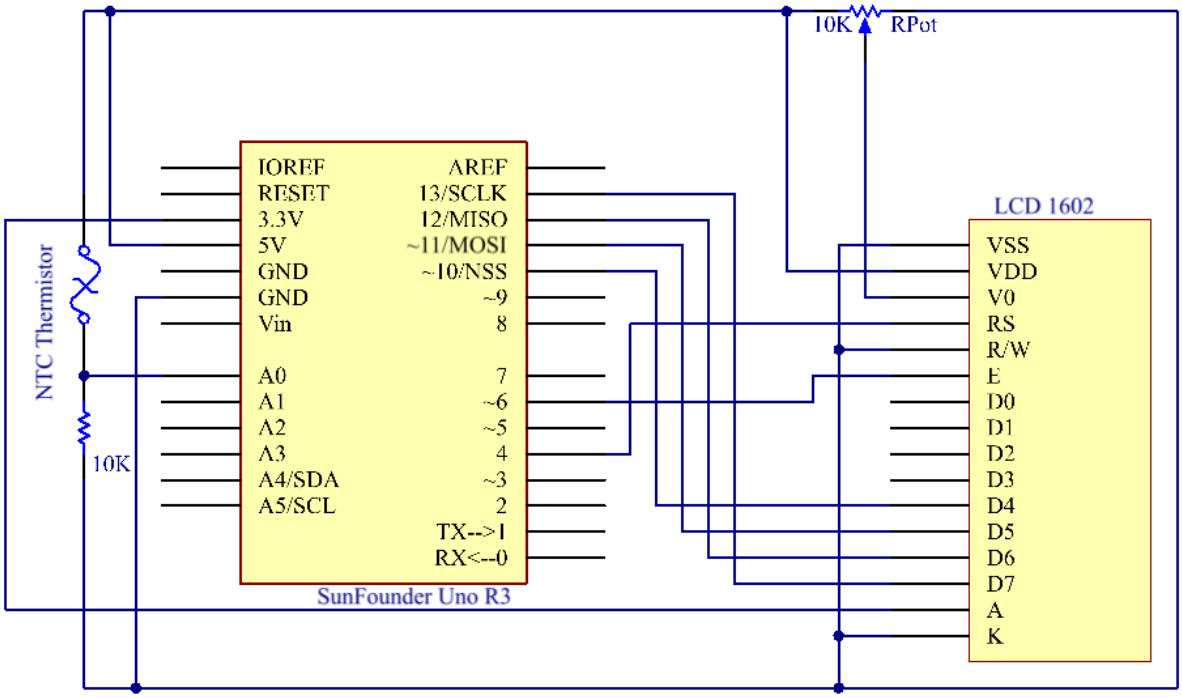
- *LCD1602 液晶显示屏*

6.12.3 原理图

热敏电阻是一种敏感元件，它有两种类型：负温度系数（NTC）和正温度系数（PTC），也有 NTC 和 PTC。其电阻随温度显着变化。当 NTC 的电阻降低时，PTC 热敏电阻的电阻随着温度的升高而增加。

在这个实验中，我们使用一个 NTC。

原理图如下所示：



其原理是 NTC 热敏电阻的阻值随着外界环境的温差而变化。它检测环境的实时温度。当温度升高时，热敏电阻的阻值减小，A0 脚电压相应升高。然后电压数据由 A/D 适配器转换为数字量。然后通过编程输出摄氏和华氏温度，然后显示在 LCD1602 上。

在这个实验中，使用了一个热敏电阻和一个 10k 的上拉电阻。每个热敏电阻都有一个正常的电阻。这里是 10k ohm，这是在 25 摄氏度下测量的。

这是电阻和温度变化之间的关系：

R T = R N exp B(1/TK −1/TN)

- `RT` 是 NTC 热敏电阻在温度为 TK 时的阻值。
- `RN` 是 NTC 热敏电阻在额定温度为 TN 的阻值。
- `TK` 是开尔文温度，单位是 K。
- `TN` 是额定开尔文温度；单位也是 K。
- 并且 β，这里是 NTC 热敏电阻的材料常数，也称为热敏指数。
- `exp` 是指数的缩写，是一个以 e 为基数的指数，它是一个自然数，大约等于 2.7。

请注意，此关系是一个经验公式。只有当温度和电阻在有效范围内时才准确。

由于 TK = T+273，T 为摄氏温度，电阻与温度变化的关系可转化为：

R =R o exp B[1/(T+273) –1/(To+273)]

B 是 beta 的缩写，是一个常数。这里是 4090。R o 是 10k 欧姆，T o 是 25 摄氏度。数据可以在热敏电阻的数据表中找到。同样，上述关系可以转化为一个来评估温度：

T= B/[ln(R/ 10) + (B/ 298)] –273（所以 ln 在这里表示自然对数，以 e 为底的对数）

如果我们使用固定电阻为 10k ohms 的电阻，我们可以用这个公式计算模拟输入引脚 A0 的电压：

V =10k x 5/(R+10K)

所以，可以形成这种关系：

R = (5 x 10k /V) - 10k

A0 的电压通过 A/D 适配器转换成数字 a。

a=V x (1024/5)

V=a/205

然后用表达式替换关系式 R = (5 x 10k /V) - 10k 中的 V，我们可以得到：R=1025 x 10k/a-10k。

最后将这里的公式中的 R 代入 T= B/[ln(R/ 10) + (B/ 298)] –273，就是刚刚形成的。然后我们最终得到温度的关系如下：

T = B/[ln{[1025 X 10/a) - 10]/10} 十 (B/298)] –273

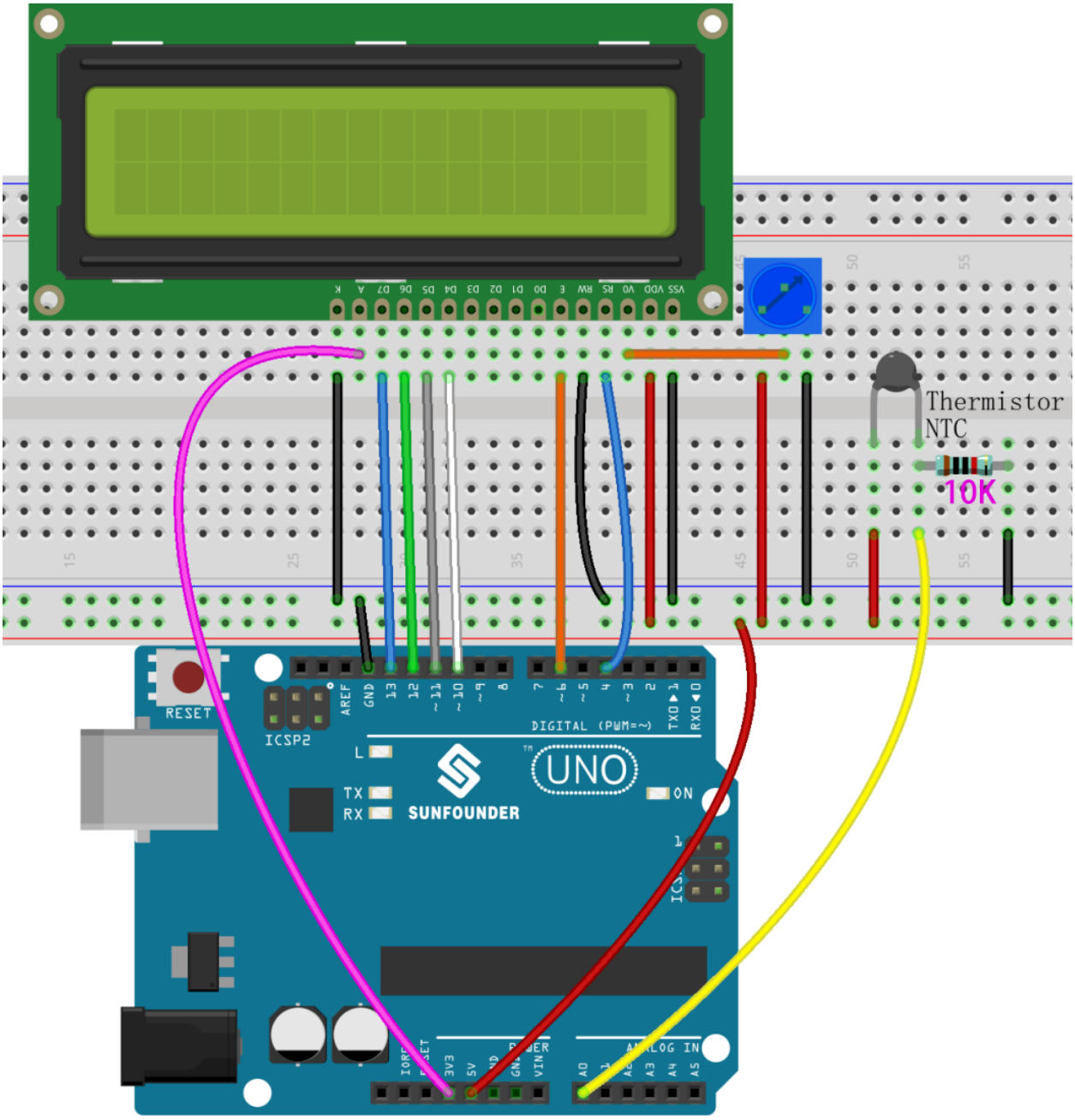
6.12.4 实验步骤

第 1 步：搭建电路。

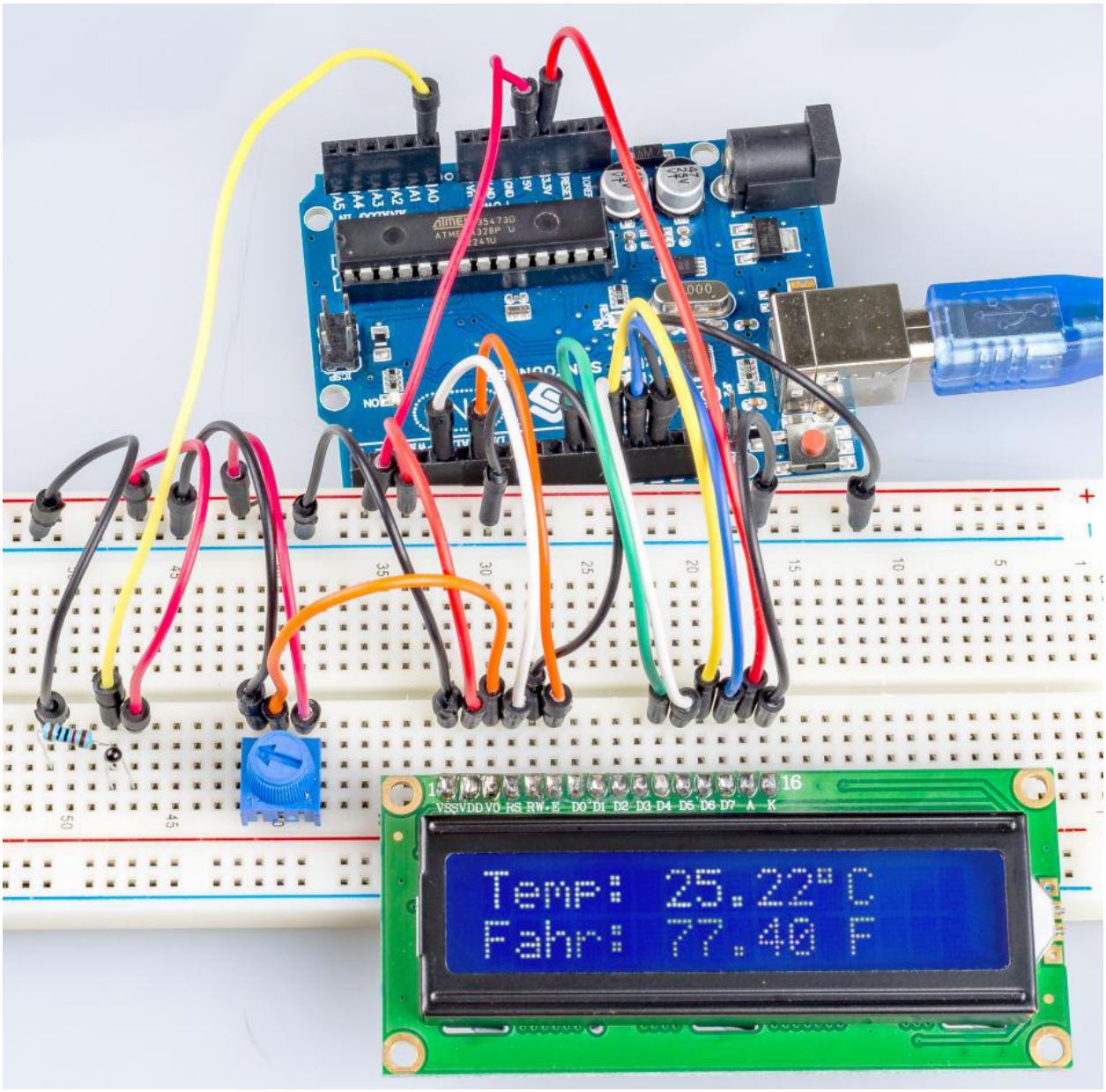
第 2 步：打开代码文件 `Lesson_12_Thermistor.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。



现在你可以在 LCD1602 上显示在摄氏度和华氏度下的温度。



6.12.5 代码

6.12.6 代码分析

设置变量

```
#define analogPin A0 //the thermistor attach to
#define beta 3950 //the beta of the thermistor
#define resistance 10 //the value of the pull-up resistor
```

设置 β 系数的值，在热敏电阻的数据表中有描述。

获取温度

```
void loop()
{
    //read thermistor value
    long a = analogRead(analogPin);
    //the calculating formula of temperature
    float tempC = beta / (log((1025.0 * 10 / a - 10) / 10) + beta / 298.0) - 273.0;
    float tempF = 1.8 * tempC + 32.0;
```

读取 A0 的值（热敏电阻），然后通过公式计算出摄氏温度，再通过公式将摄氏温度转换为华氏温度。

在 LCD1602 上显示温度

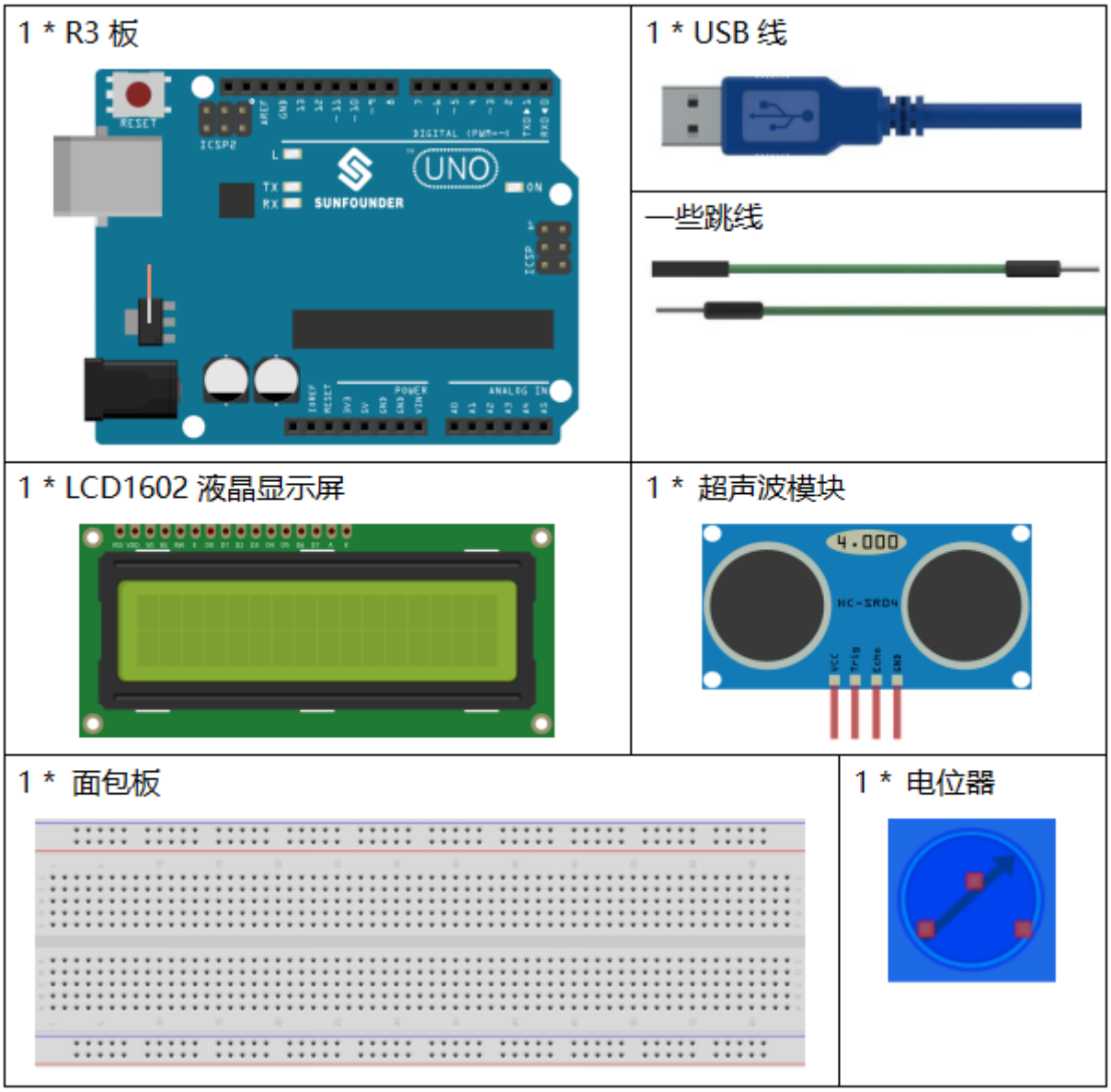
```
    lcd.setCursor(0, 0); // set the cursor to column 0, line 0
    lcd.print("Temp: ");// Print a message of "Temp: "to the LCD.
    // Print a centigrade temperature to the LCD.
    lcd.print(tempC);
    // Print the unit of the centigrade temperature to the LCD.
    lcd.print(char(223));//print the unit" ℃ "
    lcd.print("C");
    // (note: line 1 is the second row, since counting begins with 0):
    lcd.setCursor(0, 1); // set the cursor to column 0, line 1
    lcd.print("Fahr: ");
    lcd.print(tempF);// Print a Fahrenheit temperature to the LCD.
    lcd.print(" F"); // Print the unit of the Fahrenheit temperature to the LCD.
    delay(200); //wait for 100 milliseconds
}
```

6.13 第 13 课超声波

6.13.1 介绍

倒车时，你会看到汽车与周围障碍物之间的距离，以避免碰撞。检测距离的装置是超声波传感器。在本实验中，你将了解超声波如何检测距离。

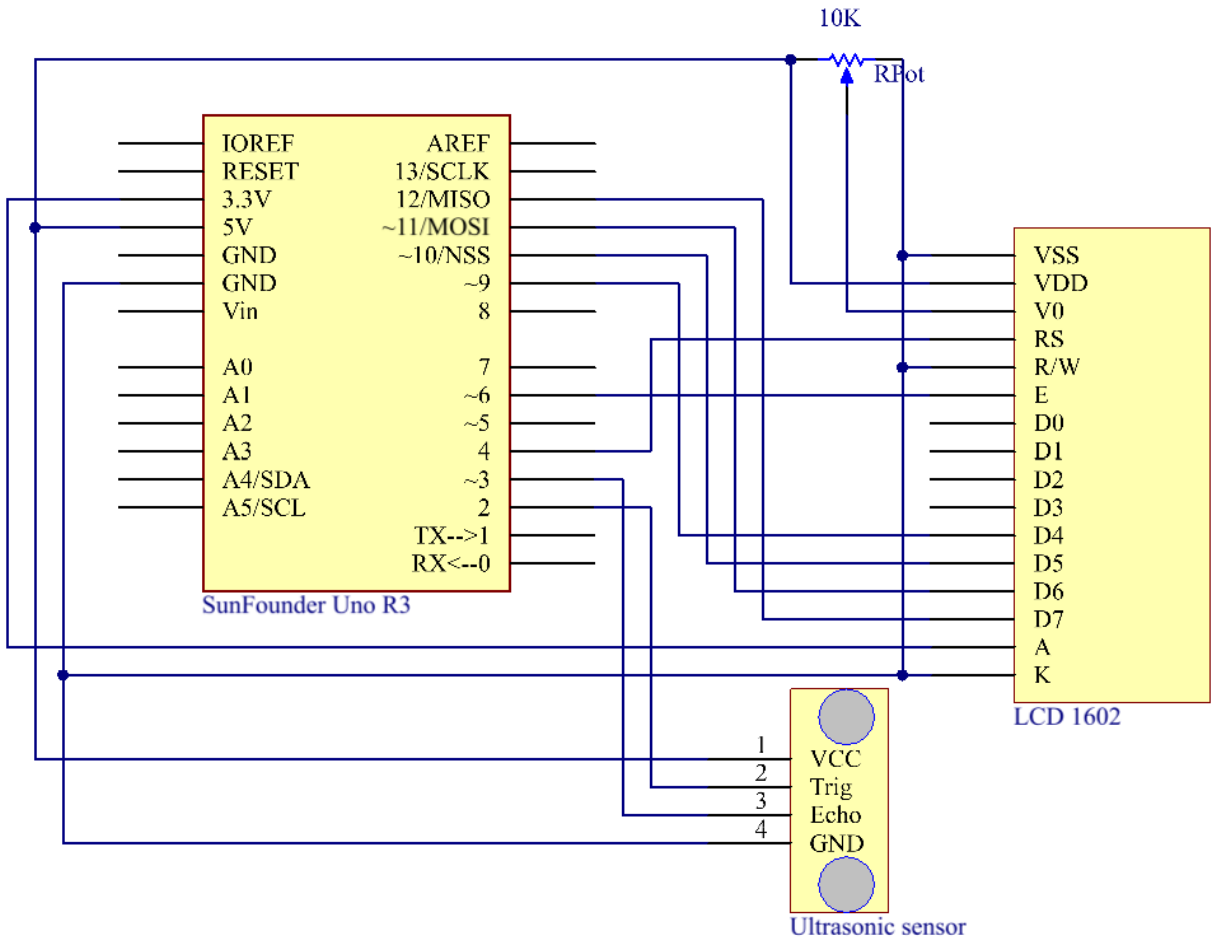
6.13.2 所需器件



- *SunFounder R3 板*
- 面包板
- 跳线
- 电位器
- 超声波模块
- *LCD1602 液晶显示屏*

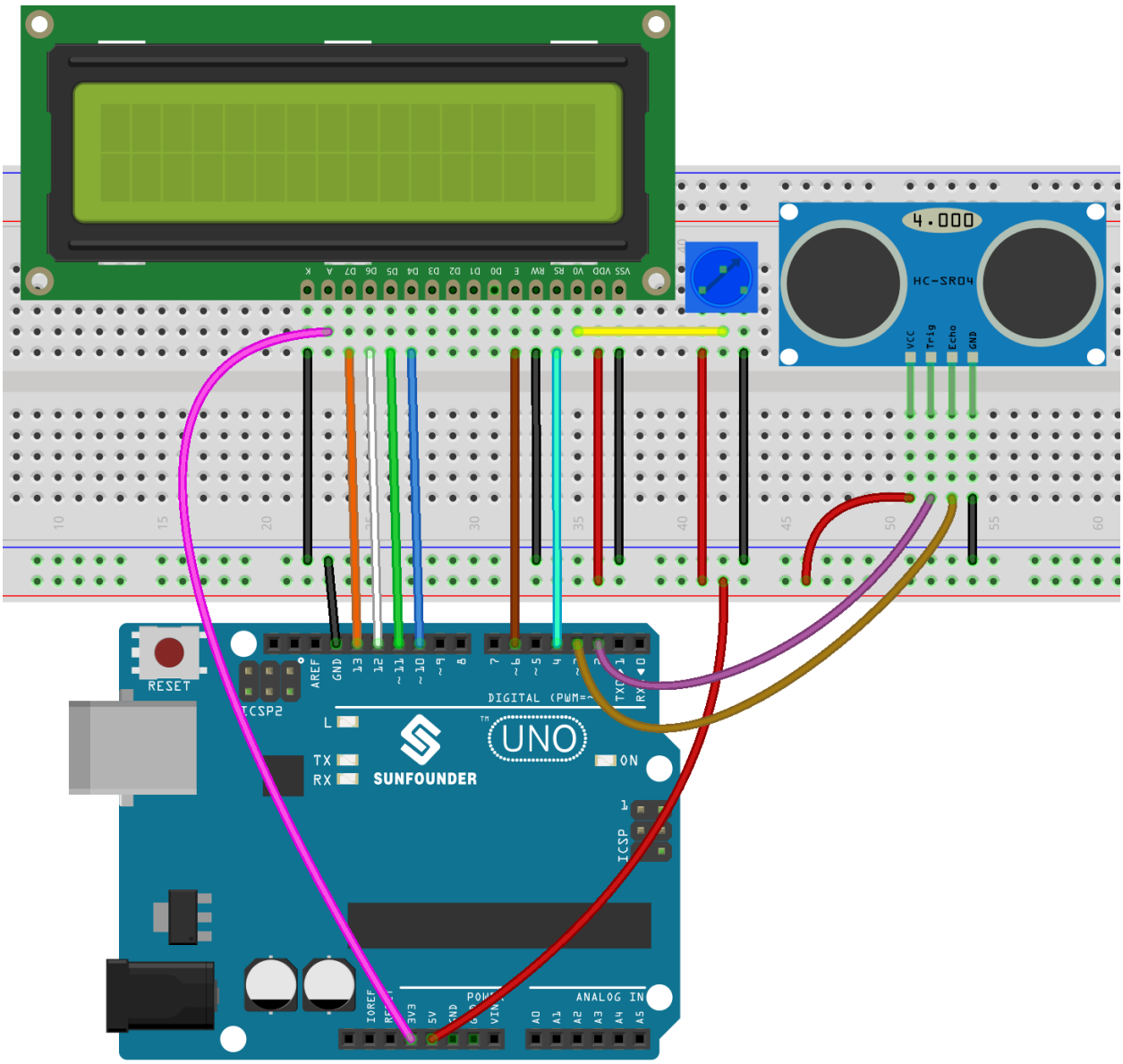
6.13.3 原理图

原理图如下所示：



6.13.4 实验步骤

第 1 步：搭建电路。



第 2 步：打开代码文件 `Lesson_13_Ultrasonic.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

备注： 如果出现如下错误，是因为你没有添加名为 NewPing 的库，请参考添加库。

```
12  #include <NewPing.h>
```

NewPing.h: No such file or directory

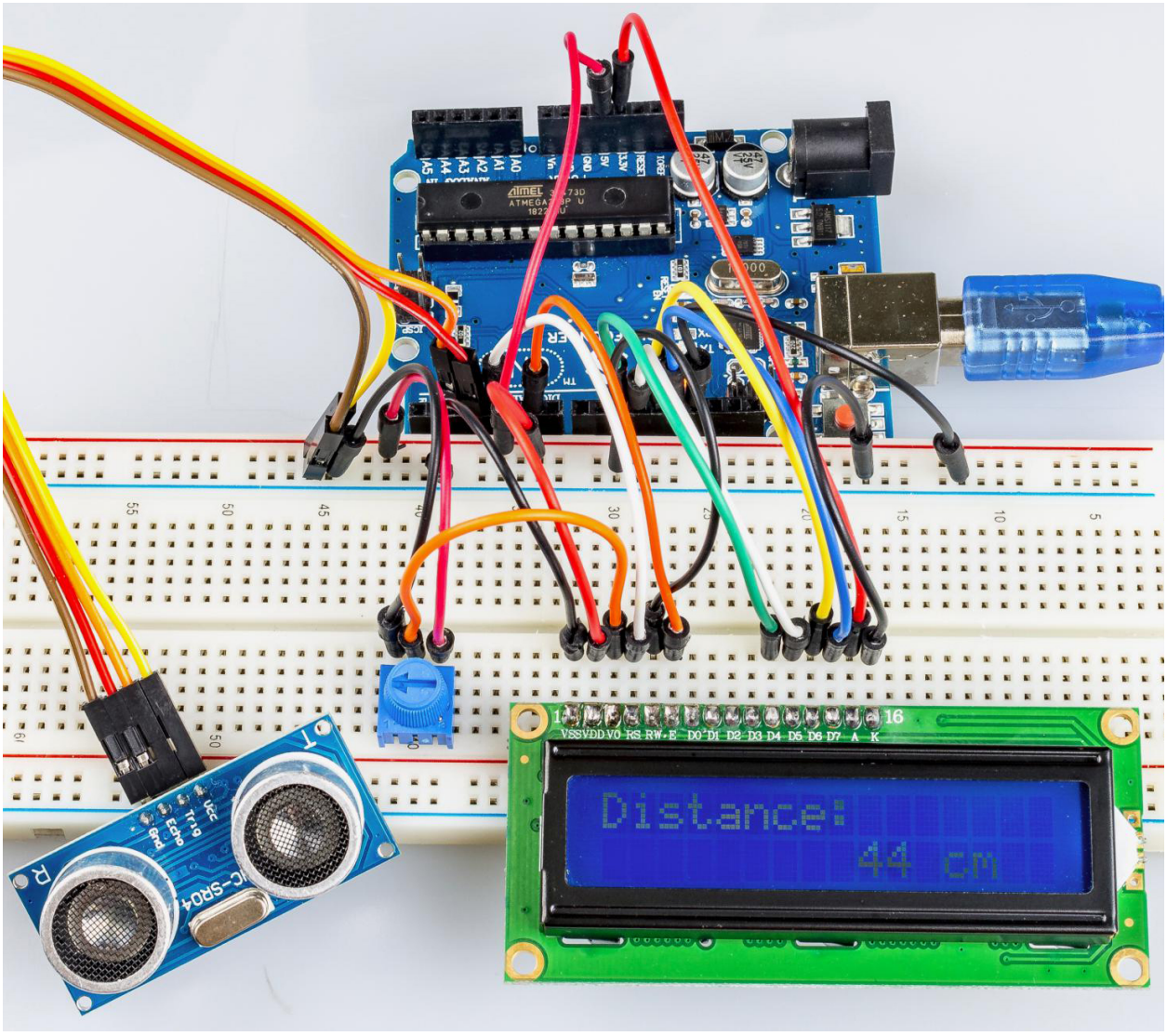
Copy error messages

```
"D:\\Program Files (x86)\\Arduino\\hardware\\tools\\avr/bin/avr-g++" -
Ultrasonic:12:21: error: NewPing.h: No such file or directory

compilation terminated.

Using library LiquidCrystal at version 1.0.7 in folder: D:\Program Fil
```

现在，如果你使用一张纸靠近或远离传感器。你会看到 LCD1602 上显示的值发生变化，这表示纸张与超声波传感器之间的距离。



6.13.5 代码

6.13.6 代码分析

初始化超声波和 LCD1602

```
#include <LiquidCrystal.h>
#include <NewPing.h>

LiquidCrystal lcd(4, 6, 10, 11, 12, 13);//lcd(RS,E,D4,D5,D6,D7)

#define TRIGGER_PIN  2  //  trig pin on the ultrasonic sensor attach to pin2 .
#define ECHO_PIN     3  //  echo pin on the ultrasonic sensor attach to pin3.
#define MAX_DISTANCE 400 // Maximum distance we want to ping for (in centimeters).␣
↪Maximum sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and␣
↪maximum distance.
```

创建一个 NewPing 变量 sonar。NewPing 的基本格式为：NewPing(uint8_t trigger_pin, uint8_t echo_pin, int max_cm_distance) 。这里 uint 表示无符号整数，8 表示 8 位。所以这里 uint8 格式的值意味着一个 unsigned-char 类型的值。

将时间转换成距离

```
unsigned int uS = sonar.ping(); // Send ping, get ping time in
microseconds (uS).
```

ping() 用来计算从脉冲发送到接收的时间。定义一个变量 Us 来存储接收的时间，单位应该是微秒（us）。

```
int distance = uS / US_ROUNDTRIP_CM;
```

uS / US_ROUNDTRIP_CM `` 是将 ``ping() 发送和接收之间的时间转换为距离的公式，单位是厘米。

在 LCD1602 上显示距离

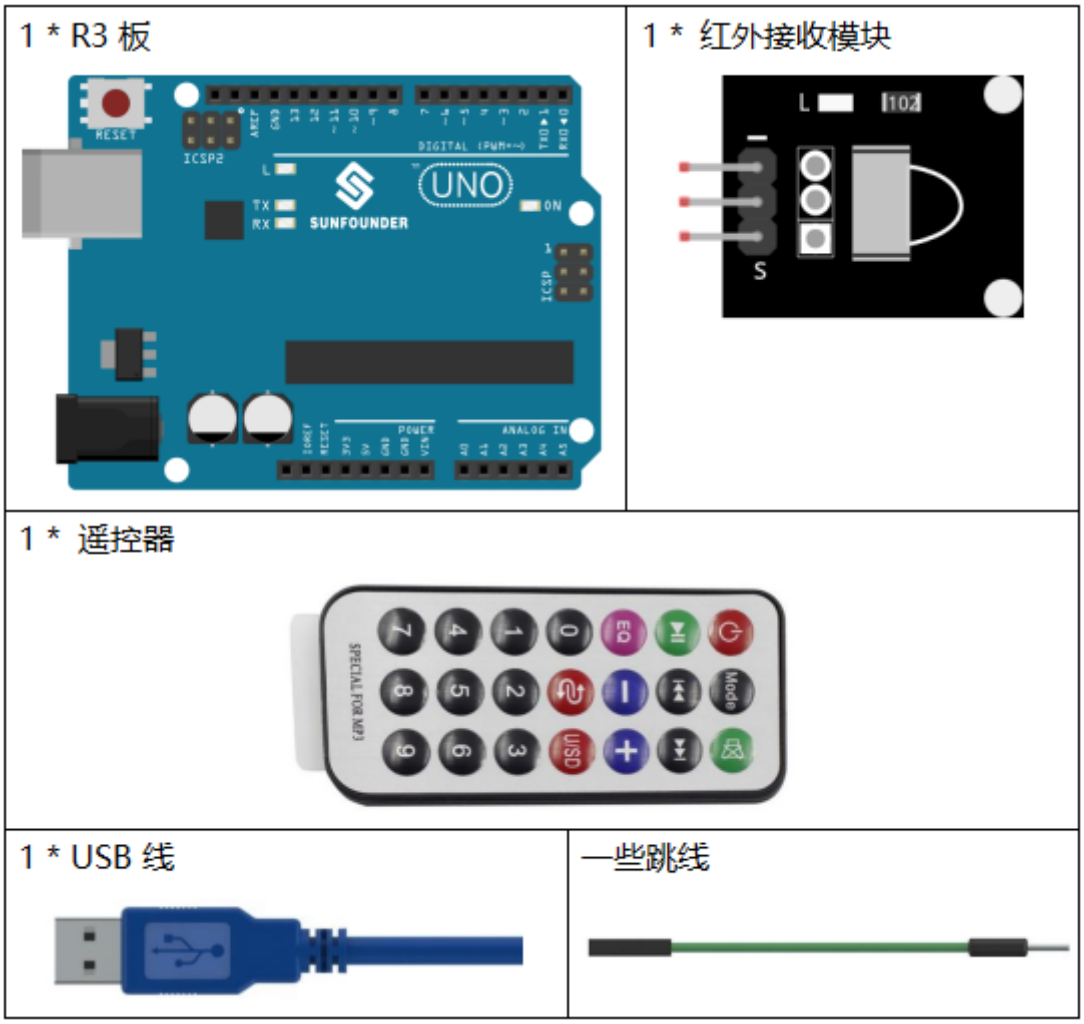
```
lcd.setCursor(0, 0);//Place the cursor at Line 1, Column 1. From here the characters␣
↪are to be displayed
lcd.print("Distance:");////Print Distance: on the LCD
lcd.setCursor(0, 1);//Set the cursor at Line 1, Column 0
lcd.print("             ");//Here is to leave some spaces after the characters so as␣
↪to clear the previous characters that may still remain.
lcd.setCursor(9, 1);//Set the cursor at Line 1, Column 9.
lcd.print(distance);// print on the LCD the value of the distance converted from the␣
↪time between ping sending and receiving.
lcd.setCursor(12, 1);//Set the cursor at Line 1, Column 12.
lcd.print("cm");//print the unit "cm"
```

6.14 第 14 课红外接收模块

6.14.1 介绍

红外接收器是接收红外信号并能独立接收红外线并输出兼容 TTL 电平的信号的部件。它的尺寸与普通的塑料封装晶体管相似，适用于各种红外遥控和红外传输。

6.14.2 所需器件

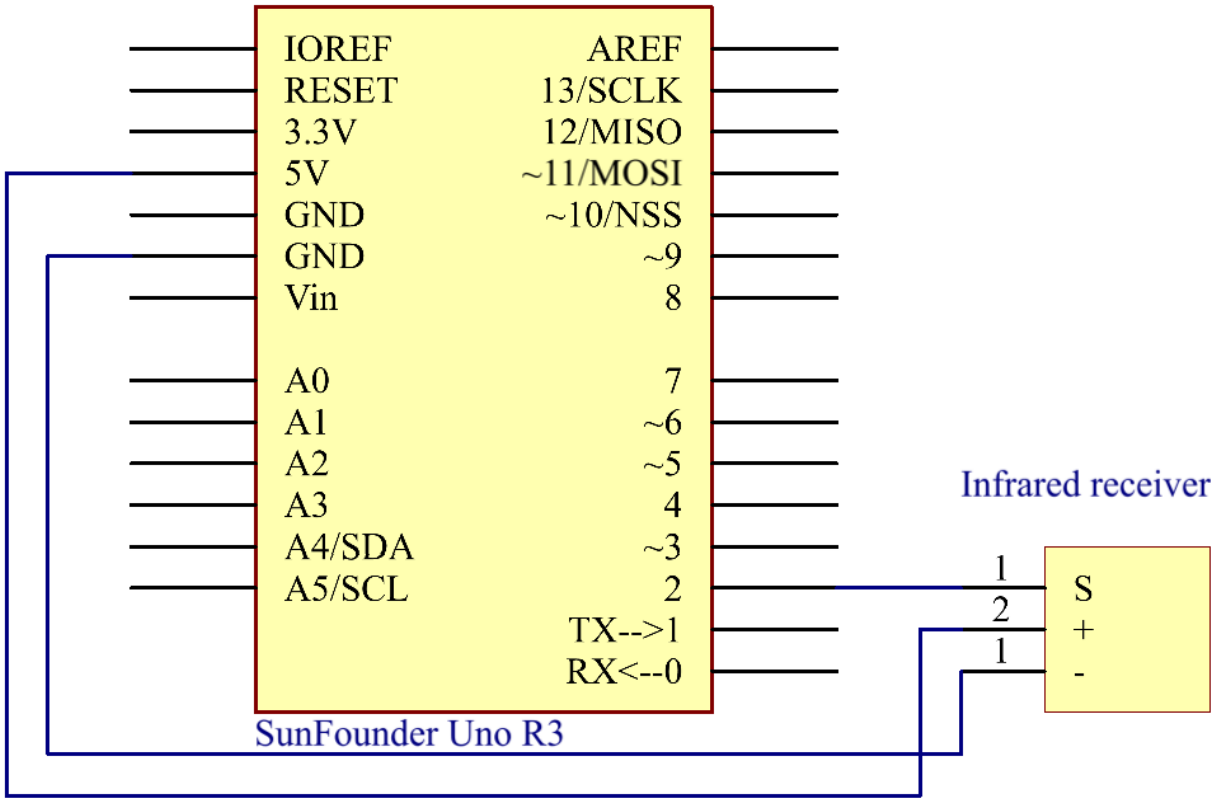


- *SunFounder R3 板*
- 面包板
- 跳线
- 红外接收模块

6.14.3 原理图

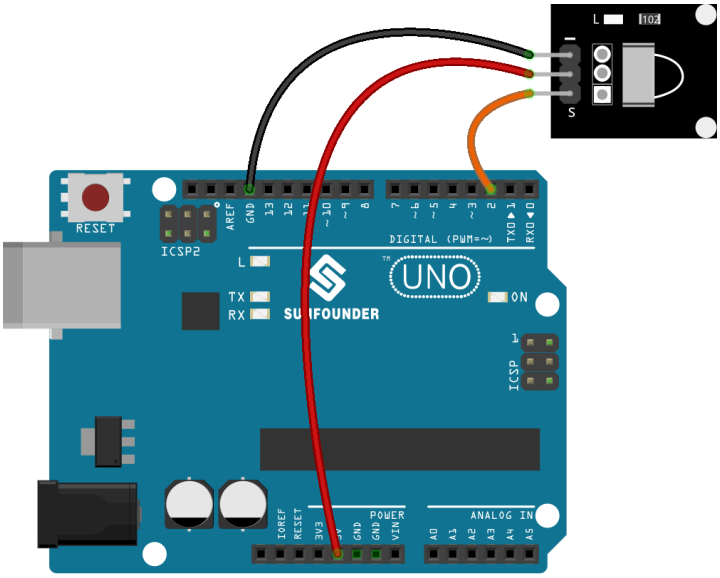
通过编程读取遥控上的某个键的键值（例如，电源键）。当你按下该键时，红外线会从遥控器发出并被红外线接收器接收，控制板上的 LED 会亮起。

原理图如下所示：



6.14.4 实验步骤

第 1 步：搭建电路。.



第 2 步：打开代码文件 `Lesson_14_Infrared_Receiver.ino`。

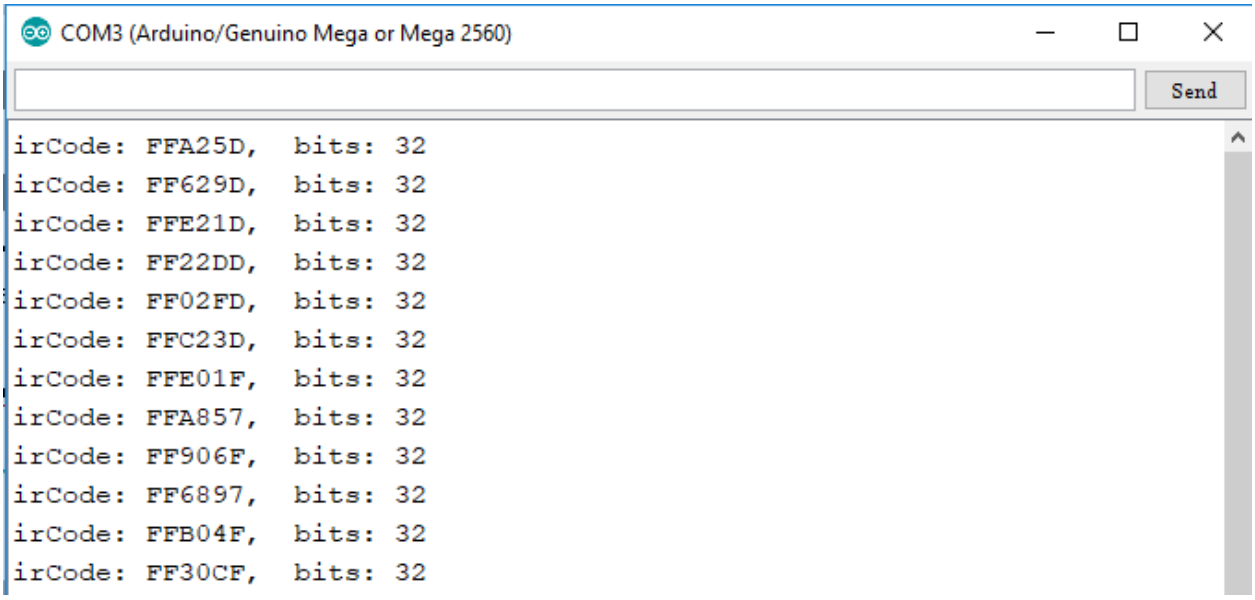
第 3 步：选择 **开发板**和 **端口**。

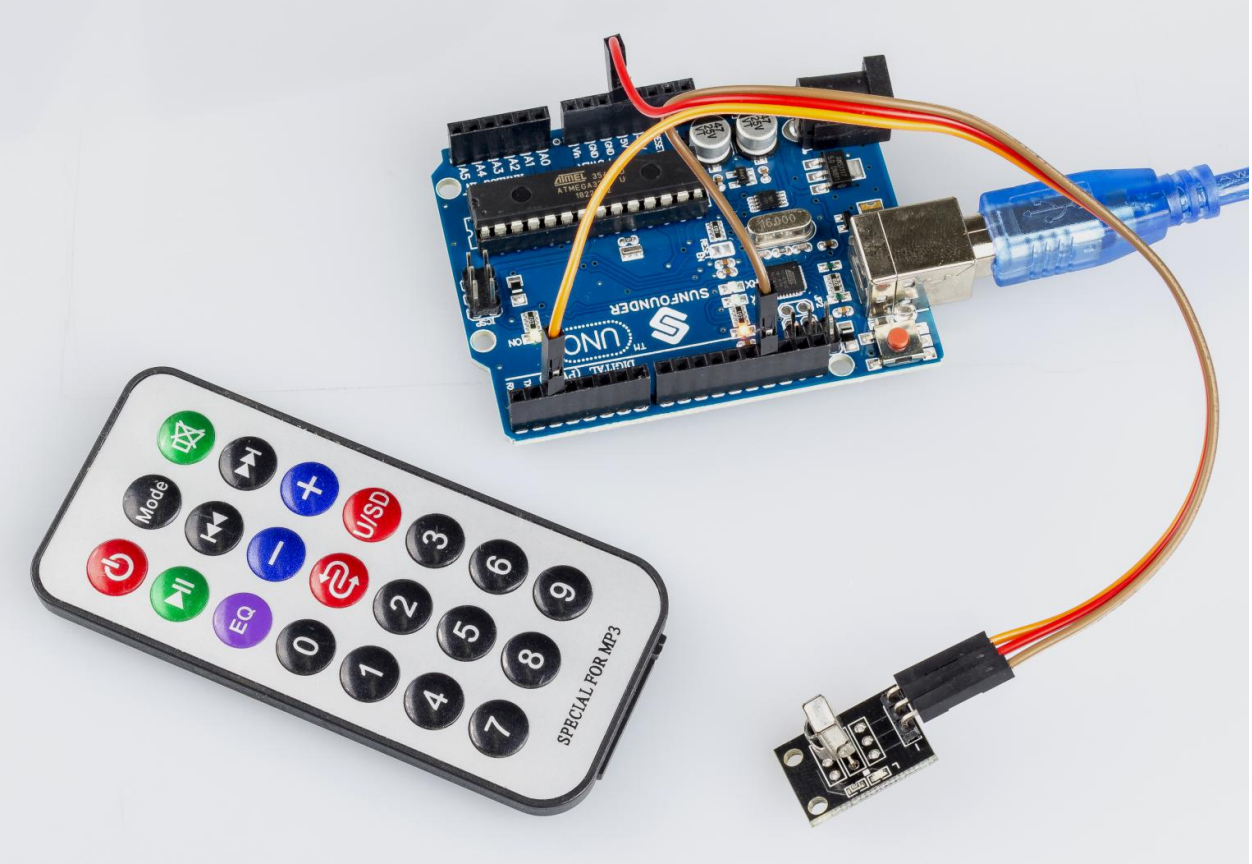
第 4 步：点击 **上传**按钮来上传代码。

现在，按下遥控器上的电源，控制板上连接到引脚 13 的 LED 将亮起。如果按其他键，LED 将熄灭。

备注:

- 遥控器的尾部有一块透明塑料片用来切断电源，你需要在使用前拔出。
- 请轻轻按下遥控器上的按钮，以避免无效数据 FFFFFFFF。





6.14.5 代码

6.14.6 代码分析

初始化红外接收器

```
#include <IRremote.h>
const int irReceiverPin = 2; // the infrared-receiver attact to pin2
const int ledPin = 13; // built-in LED
IRrecv irrecv(irReceiverPin); // Initialize the infrared-receiver
decode_results results; // The decoding result is placed in the result of the decode␣
↪results structure.
```

启用红外接收器

```
irrecv.enableIRIn(); // Restart the receiver
```

接收并打印数据

```
if (irrecv.decode(&results)) { // If receive a data
```

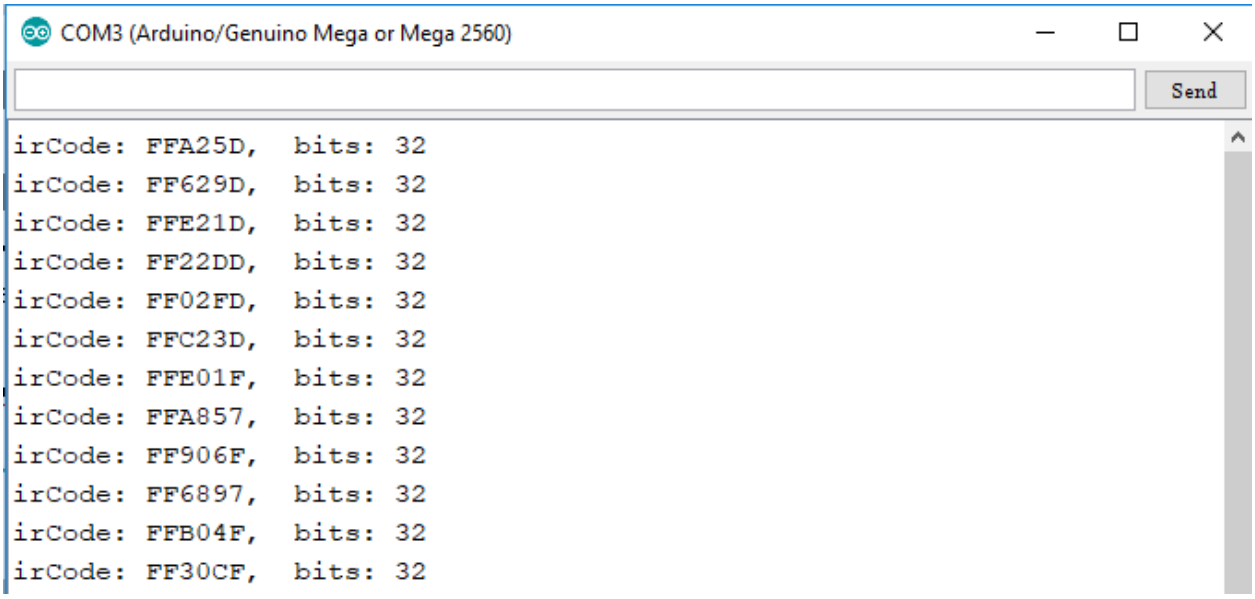
`decode(&results)`：对接收到的红外信息进行解码，没有数据返回 0，否则返回 1。解码结果存放在 `results` 中。

```
    Serial.print("irCode: "); // print "irCode: " on the serial monitor
    Serial.print(results.value, HEX); // print the signal on serial monitor
    in hexadecimal
    Serial.print(", bits: ");
    Serial.println(results.bits); // Print the data bits
    irrecv.resume(); // Receive next data
}
delay(600);
```

如果电源键被按下

```
if(results.value == 0xFFA25D) // if the power button on the remote control is pressed
```

0xFFA25D 是遥控器电源键的代码，如果你想定义其他按钮，你可以从串口监视器上读取每个按键的代码。




```
{
    digitalWrite(ledPin,HIGH); // Turn on the LED
}
else
{
    digitalWrite(ledPin,LOW); // else turn of the LED
}
```

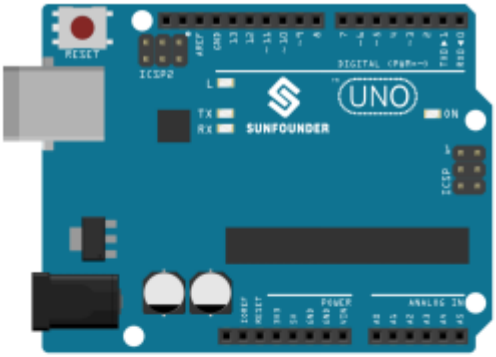
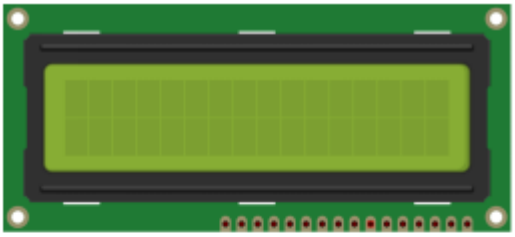
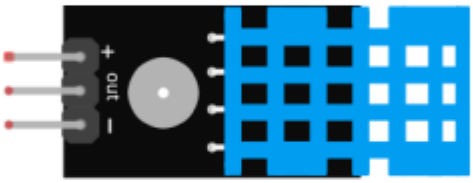
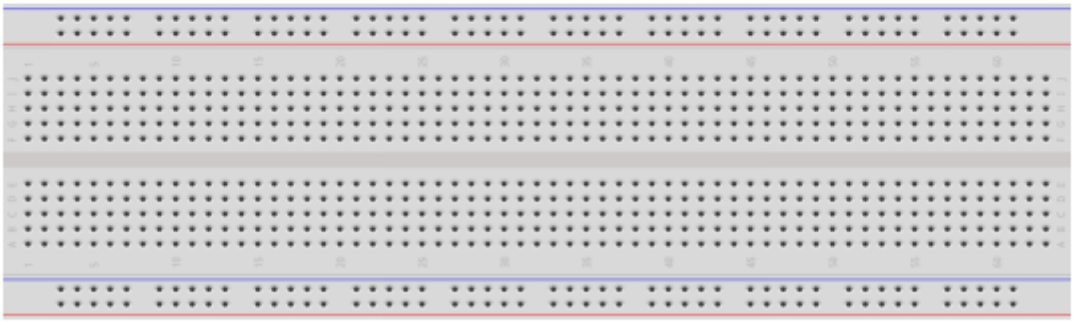
6.15 第 15 课温湿度传感器

6.15.1 介绍

数字温湿度传感器 DHT11 是一种复合传感器，包含经过校准的温湿度数字信号输出。采用专用数字模块采集技术和温湿度传感技术，确保产品具有高可靠性和优异的长期稳定性。

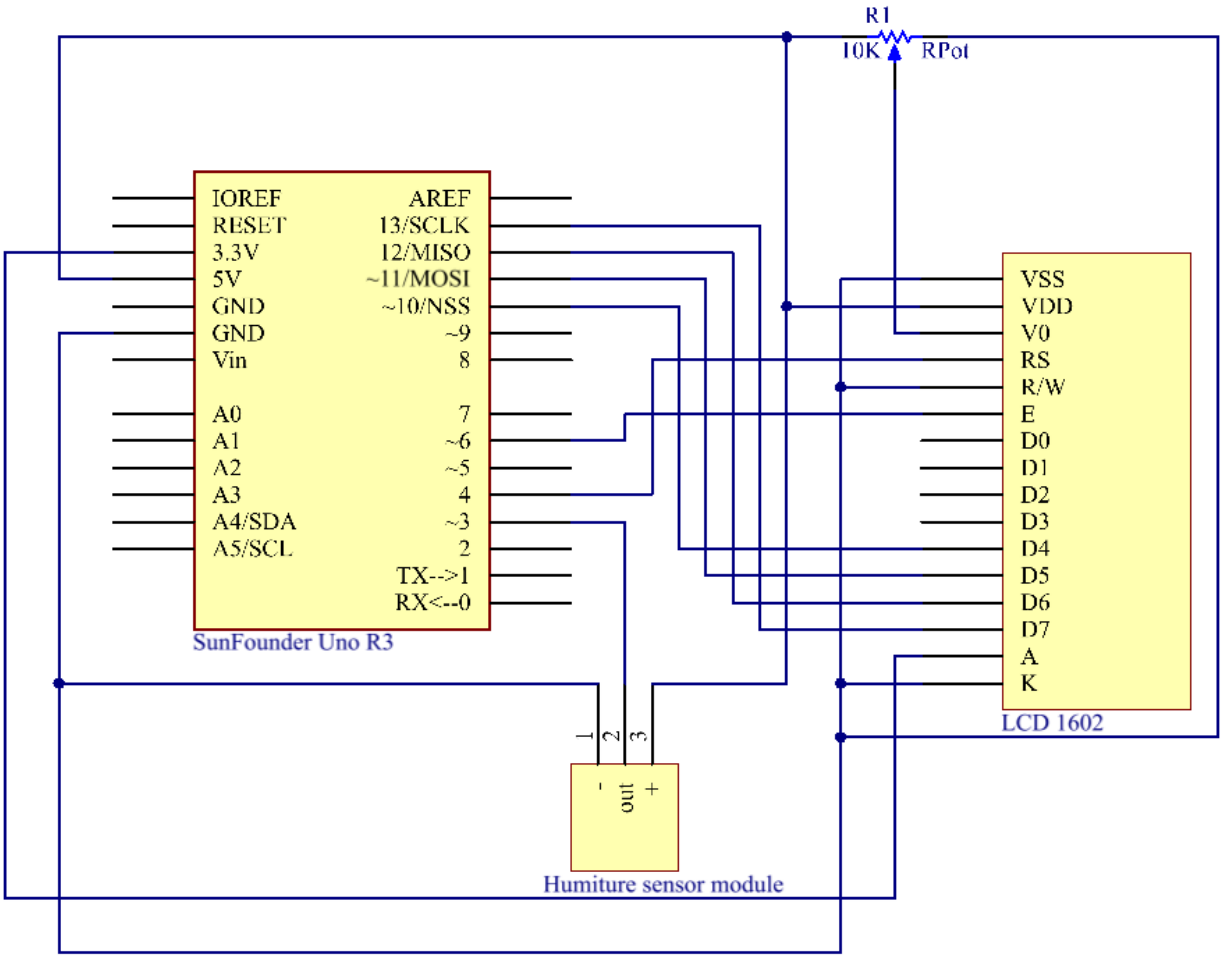
该传感器包括电阻感湿元件和 NTC 测温装置，并与高性能 8 位微控制器相连。

6.15.2 所需器件

1 * R3 板 	1 * LCD1602 液晶显示屏 
1 * 电位器 	1 * 温湿度传感器模块 
1 * 面包板 	
1 * USB 线 	一些跳线 

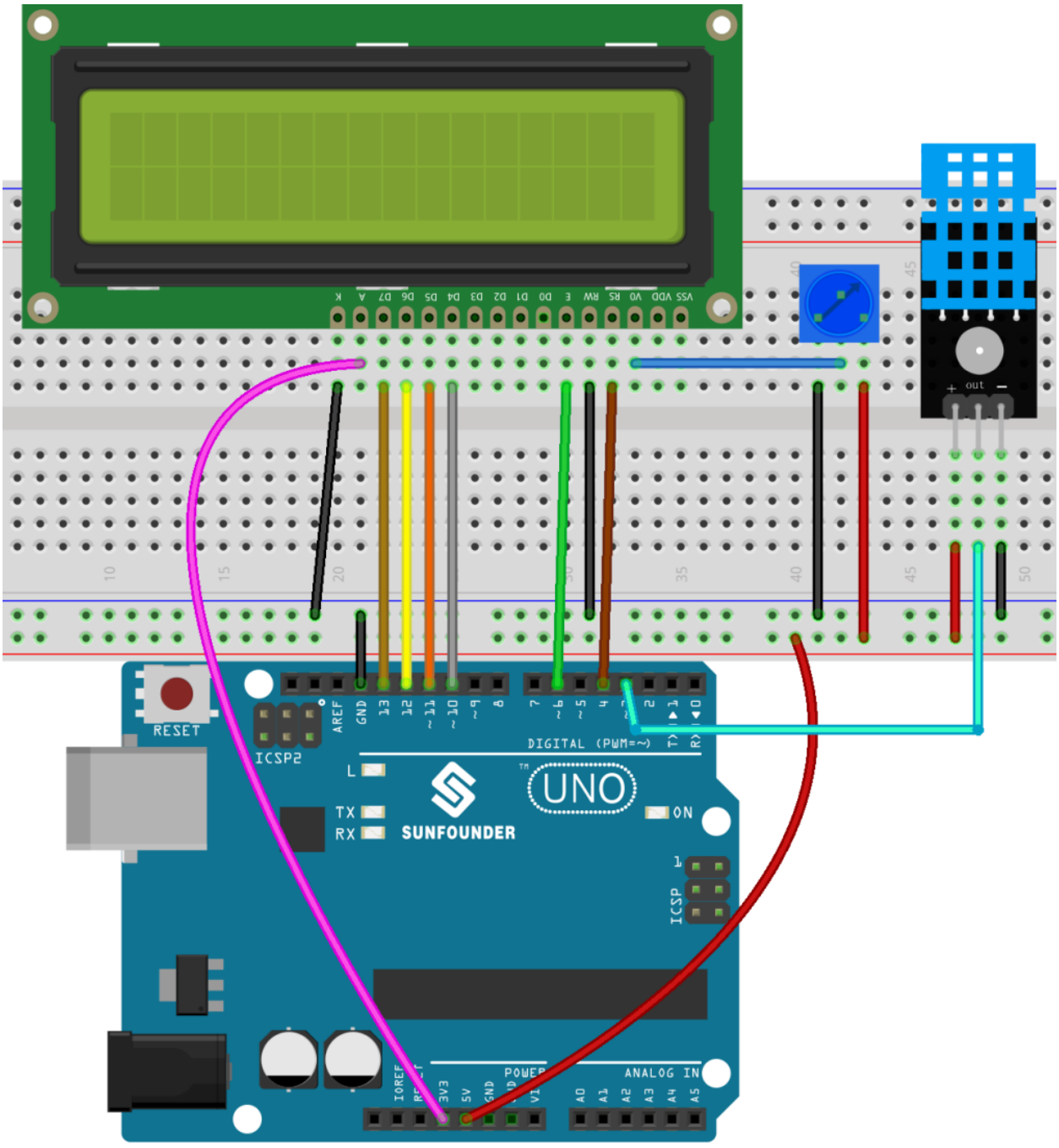
- *SunFounder R3 板*
- *面包板*
- *跳线*
- *LCD1602 液晶显示屏*
- *电位器*
- *温湿度传感器模块*

6.15.3 原理图



6.15.4 实验步骤

第 1 步：搭建电路。.

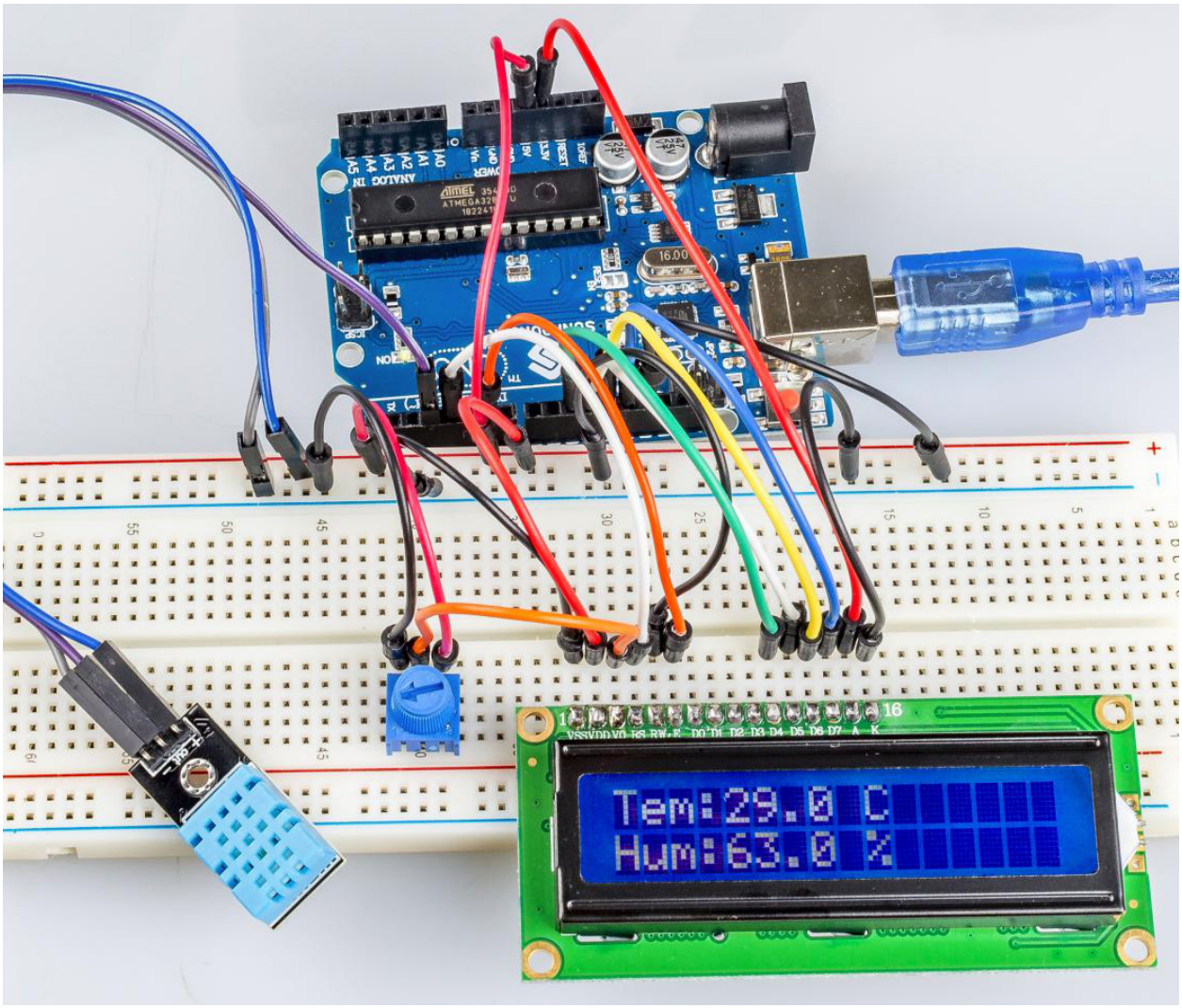


第 2 步：打开代码文件 `Lesson_15_Humiture_Sensor.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你可以看到 LCD1602 上显示的当前湿度和温度值。



6.15.5 代码

6.15.6 代码分析

初始化温湿度和 LCD1602

```
#include <dht.h> // Include the head file dht.h
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 6, 10, 11, 12, 13); // initialize the LCD1602
dht DHT;
#define DHT11_PIN 3 // the humiture sensor attact to pin3
```

读温湿度传感器的值

```
int chk = DHT.read11(DHT11_PIN);
switch (chk)
{
    case DHTLIB_OK:
        Serial.println("OK,\t");
```

(续下页)

(接上页)

```
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.println("Checksum error,\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.println("Time out error,\t");
        break;
    default:
        Serial.println("Unknown error,\t");
        break;
}
```

使用该 `read11()` 函数读取温湿度传感器的值。如果串口监视器上显示 OK，则说明温湿度传感器工作正常。

- `read11()`：返回值：

```
// DHTLIB_OK: Indicate the humiture sensor is work well.
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
```

LCD1602 上的显示

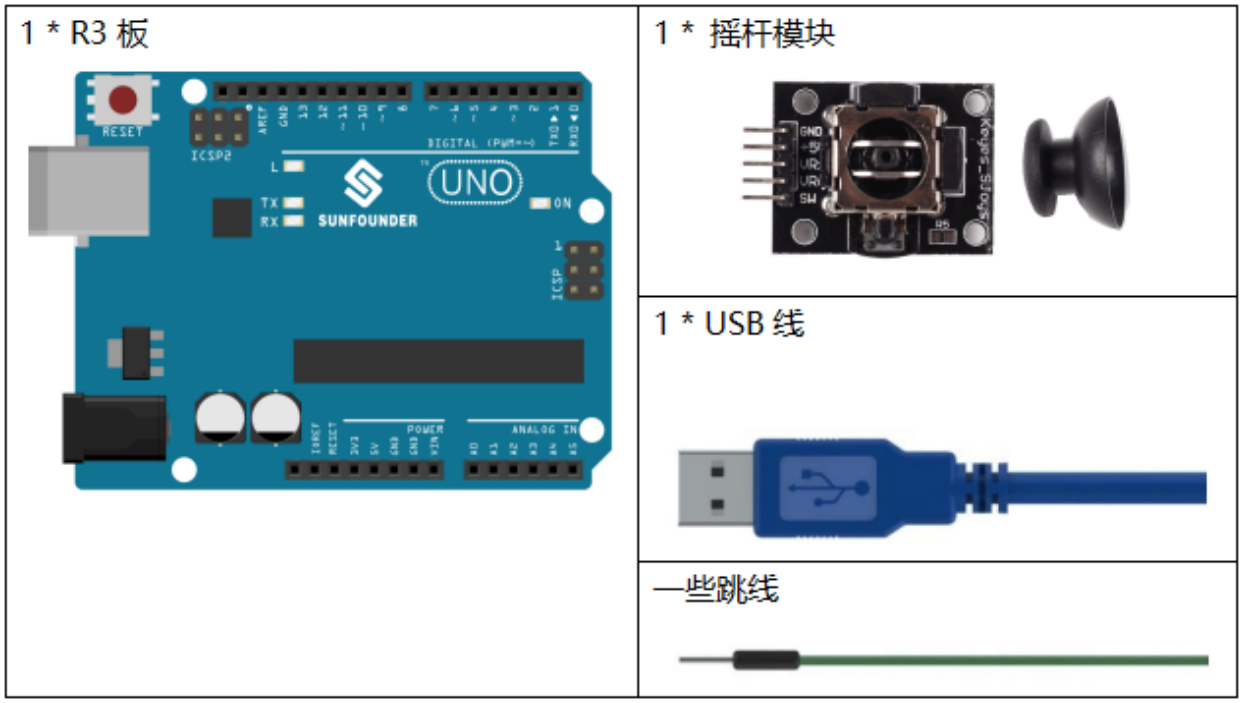
```
lcd.setCursor(0, 0);
lcd.print("Tem:");
lcd.print(DHT.temperature,1); //print the temperature on lcd
lcd.print(" C");
lcd.setCursor(0, 1);
lcd.print("Hum:");
lcd.print(DHT.humidity,1); //print the humidity on lcd
lcd.print(" %");
delay(200); //wait a while
```

6.16 第 16 课摇杆

6.16.1 介绍

操纵杆是一种输入设备，由一个在底座上旋转的操纵杆组成，并向它所控制的设备报告其角度或方向。操纵杆通常用于控制视频游戏和机器人，此处使用操纵杆 PS2。

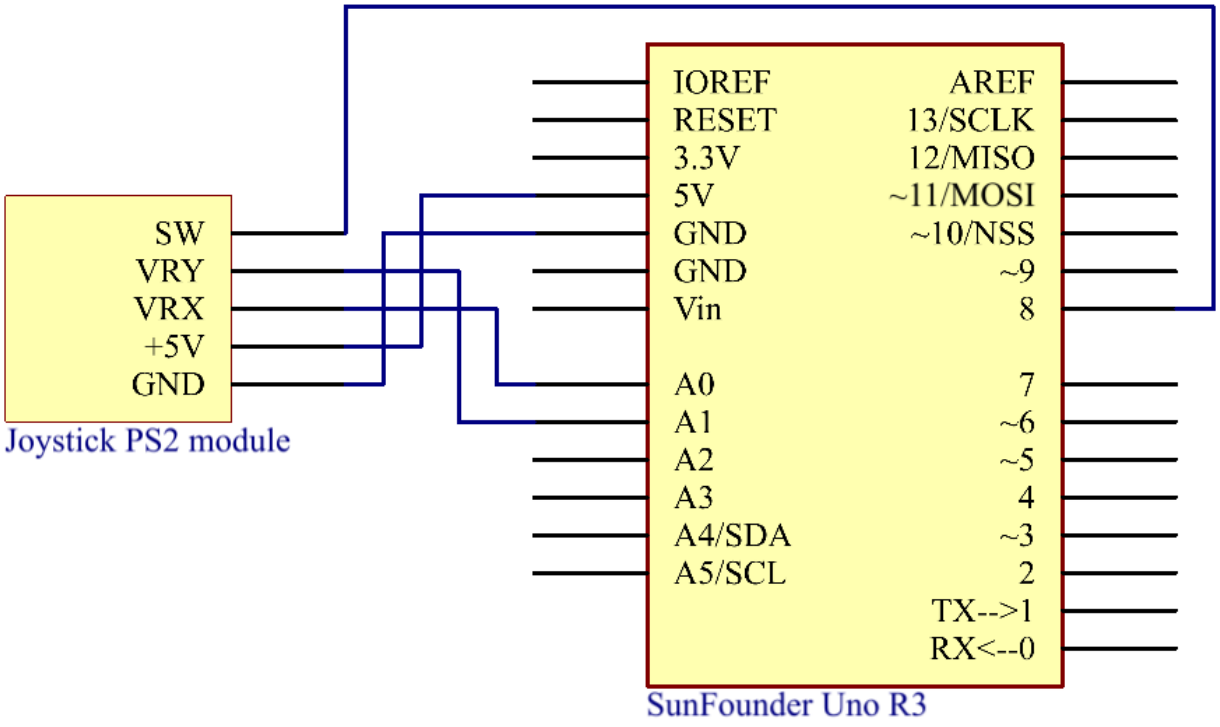
6.16.2 所需器件



- *SunFounder R3 板*
- 面包板
- 跳线
- 摇杆模块

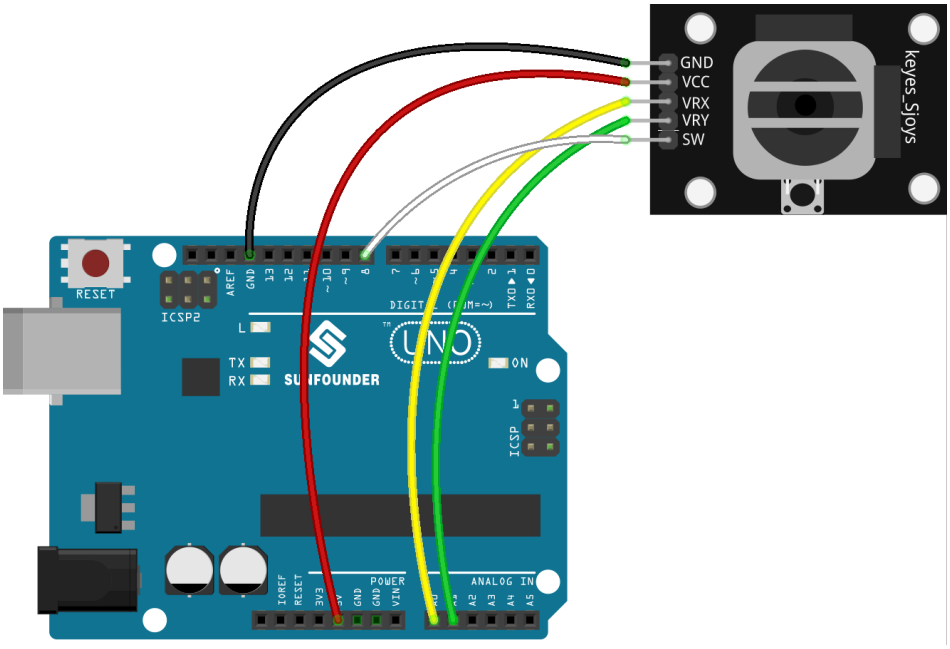
6.16.3 原理图

该模块有 2 路模拟量输出（对应 X，Y 双轴偏移）和 1 路数字输出（Z 轴），用来表示摇杆是否被按压。
在这个实验中，将在串口监视器中显示摇杆上的 X,Y 和 Z 轴上的值的变化。
原理图如下所示：



6.16.4 实验步骤

第 1 步：搭建电路。

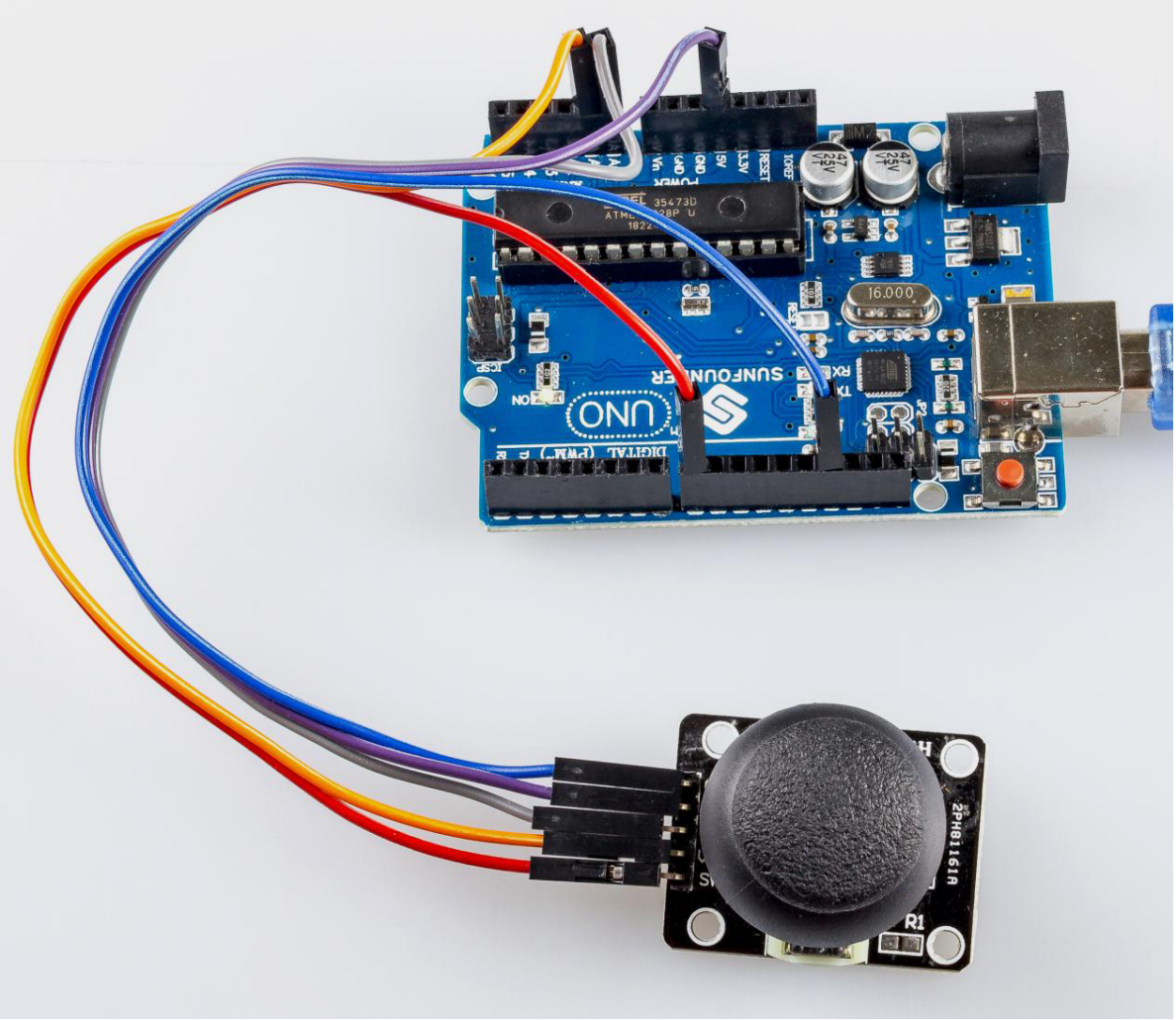


第 2 步：打开代码文件 `Lesson_16_Joystick_PS2.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，拨动摇杆，串口监视器上显示的 X 和 Y 轴坐标会相应改变；按摇杆，会显示 Z=0。



6.16.5 代码

6.16.6 代码分析

该代码使用串行监视器打印操纵杆 ps2 的 VRX、VRY 和 SW 引脚的值。

```
void loop()
{
    Serial.print("X: ");
    Serial.print(analogRead(xPin), DEC);  // print the value of VRX in DEC
    Serial.print("|Y: ");
    Serial.print(analogRead(yPin), DEC);  // print the value of VRX in DEC
    Serial.print("|Z: ");
    Serial.println(digitalRead(swPin));   // print the value of SW
```

(续下页)

(接上页)

```
    delay(50);
}
```

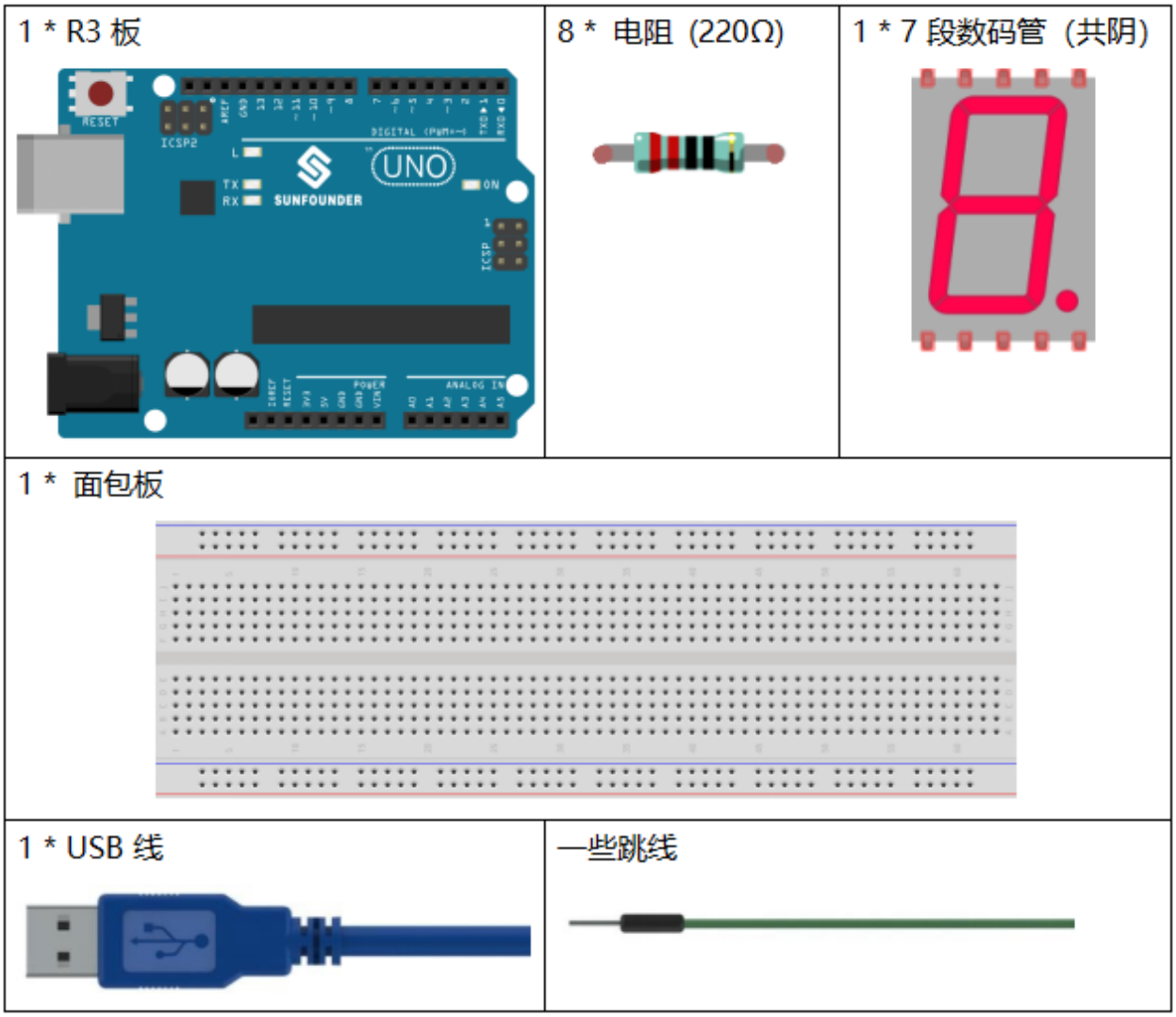
- "|Y: " 中的 | 用来隔开数据。

6.17 第 17 课 7 段数码管

6.17.1 介绍

7 段数码管是一种可以显示数字和字母的设备。它由七个并联的 LED 组成。通过将数码管上的引脚连接到电源并启用相关引脚，从而打开相应的 LED 段，可以显示不同的字母/数字。在本课中，我们将学习如何在其上显示特定字符 。

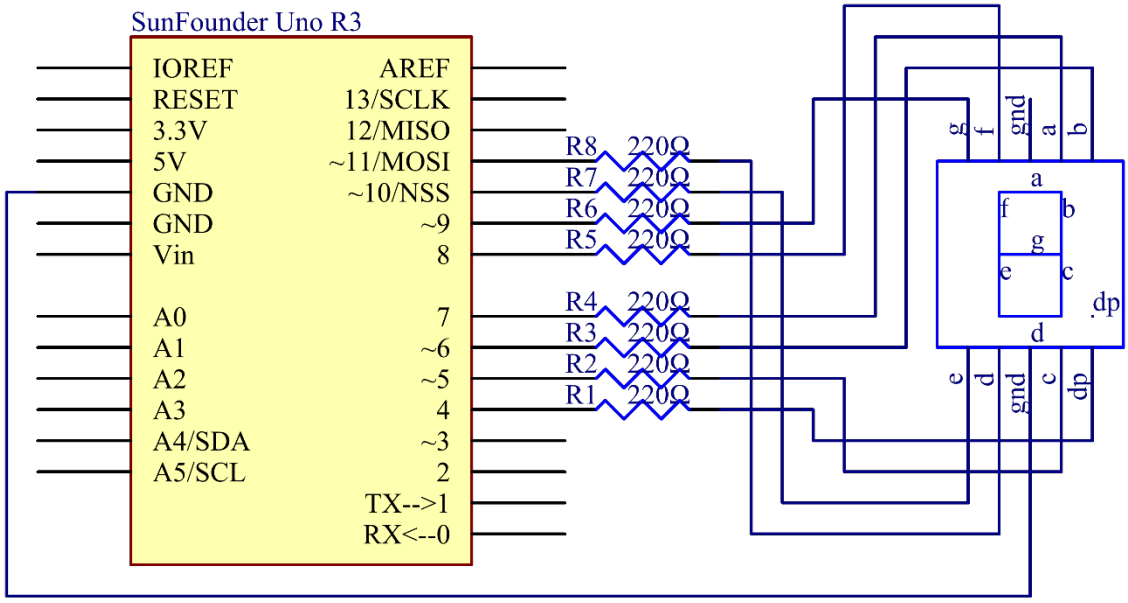
6.17.2 所需器件



- *SunFounder R3 板*
- 面包板
- 跳线
- 电阻
- *7 段数码管*

6.17.3 原理图

在本实验中，将 7 段数码管的每个引脚 a~g 分别连接到一个 220 欧姆的限流电阻，然后连接到引脚 4-11。GND 连接到 GND。通过编程，我们可以将 pin4-11 中的一个或几个设置为高电平来点亮相应的 LED。

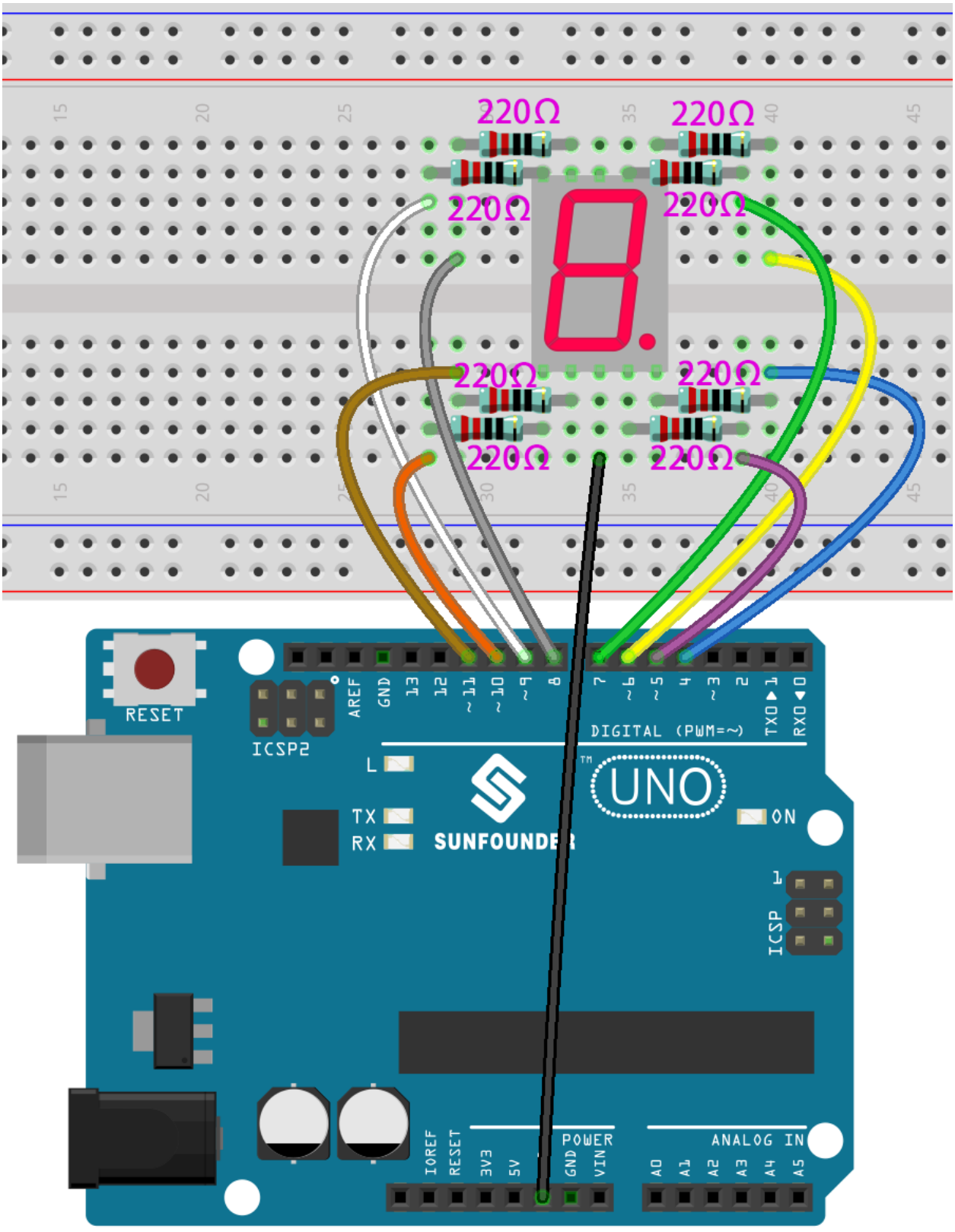


6.17.4 实验步骤

第 1 步：搭建电路 (这里使用的是共阴极 7 段数码管)。

7 段数码管与 R3 板的接线：

7-Segment	R3 板
a	7
b	6
c	5
d	11
e	10
f	8
g	9
dp	4
“- “	GND

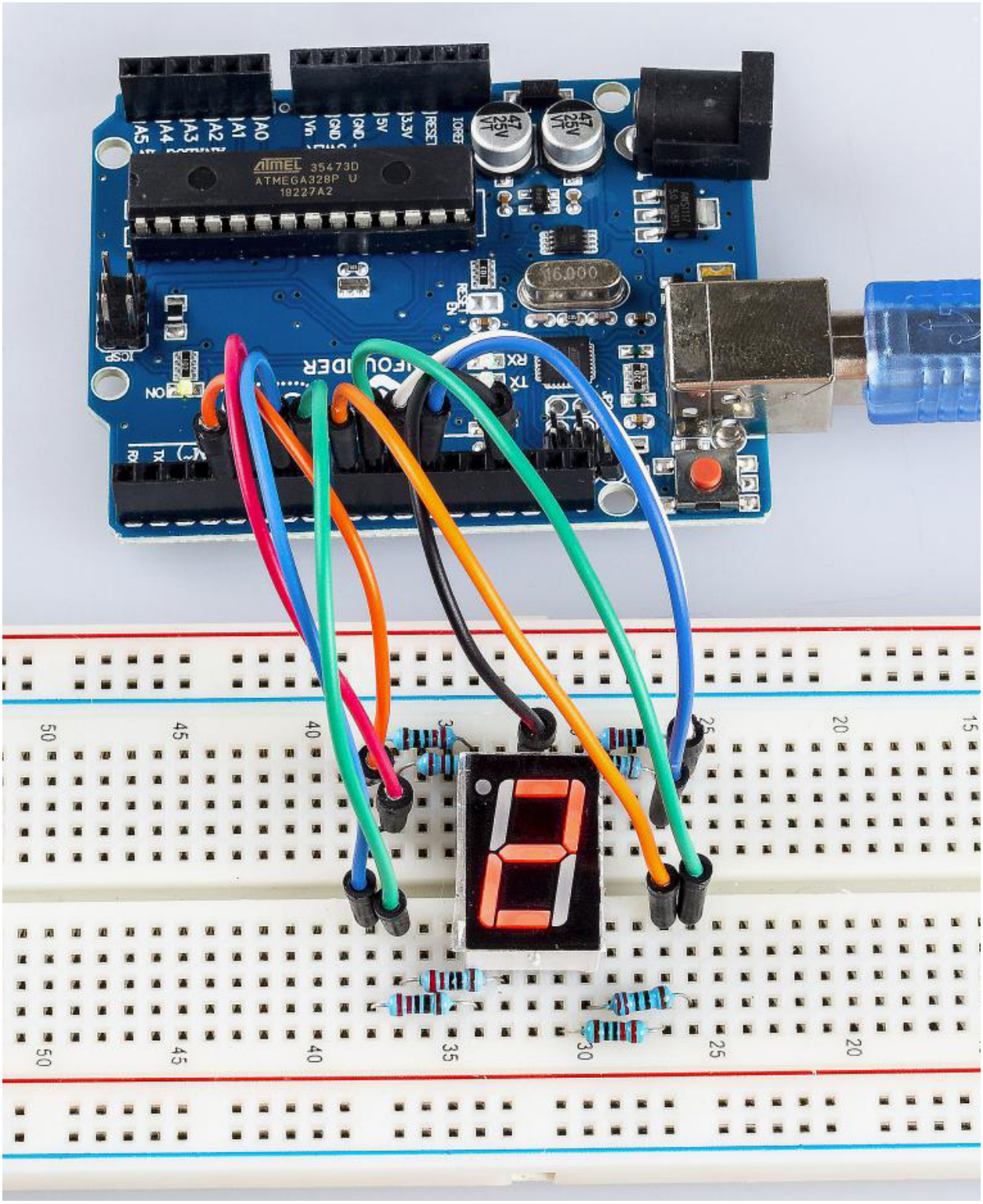


第 2 步：打开代码文件 `Lesson_17_7_Segment_Display.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

你现在应该看到 7 段数码管轮流显示 0-9，A-F。



6.17.5 代码

6.17.6 代码分析

这个实验的代码可能有点长。但是语法很简单。让我们来看看。

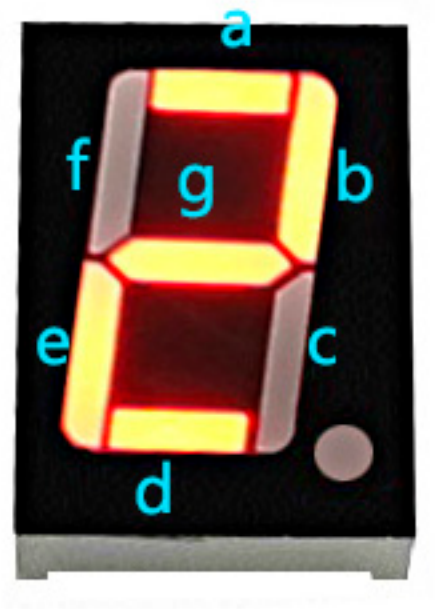
在 loop() 中调用函数

```
void loop()
{
    digital_1();//diaplay 1 to the 7-segment
    delay(1000);//wait for a second
    digital_2();//diaplay 2 to the 7-segment
    delay(1000); //wait for a second
    digital_3();//diaplay 3 to the 7-segment
    delay(1000); //wait for a second
    digital_4();//diaplay 4 to the 7-segment
...
```

将这些函数调用到 loop() 中是为了让 7 段数码管显示 0-F。功能如下所示。以 `digital_2()` 为例：

digital_2() 详解

```
void digital_2(void) //diaplay 2 to the 7-segment
{
digitalWrite(b,HIGH);
digitalWrite(a,HIGH);
for(int j = 9;j <= 11;j++)
    digitalWrite(j,HIGH);
digitalWrite(c,LOW);
digitalWrite(f,LOW);
}
```



首先我们需要知道在 7 段数码管上显示数字 2 时的样子。实际上是 a、b、d、e 和 g 段通电（被设置为高电平），c 和 f 熄灭（被设置为高电平），从而产生 2 的显示。

运行此部分后，7 段数码管将显示 2。同样，其他字符的显示也是一样的。由于大写字母 b 和 d ，即 B 和 D，在数码管上与 8 和 0 看起来相同，因此它们以小写字母显示。

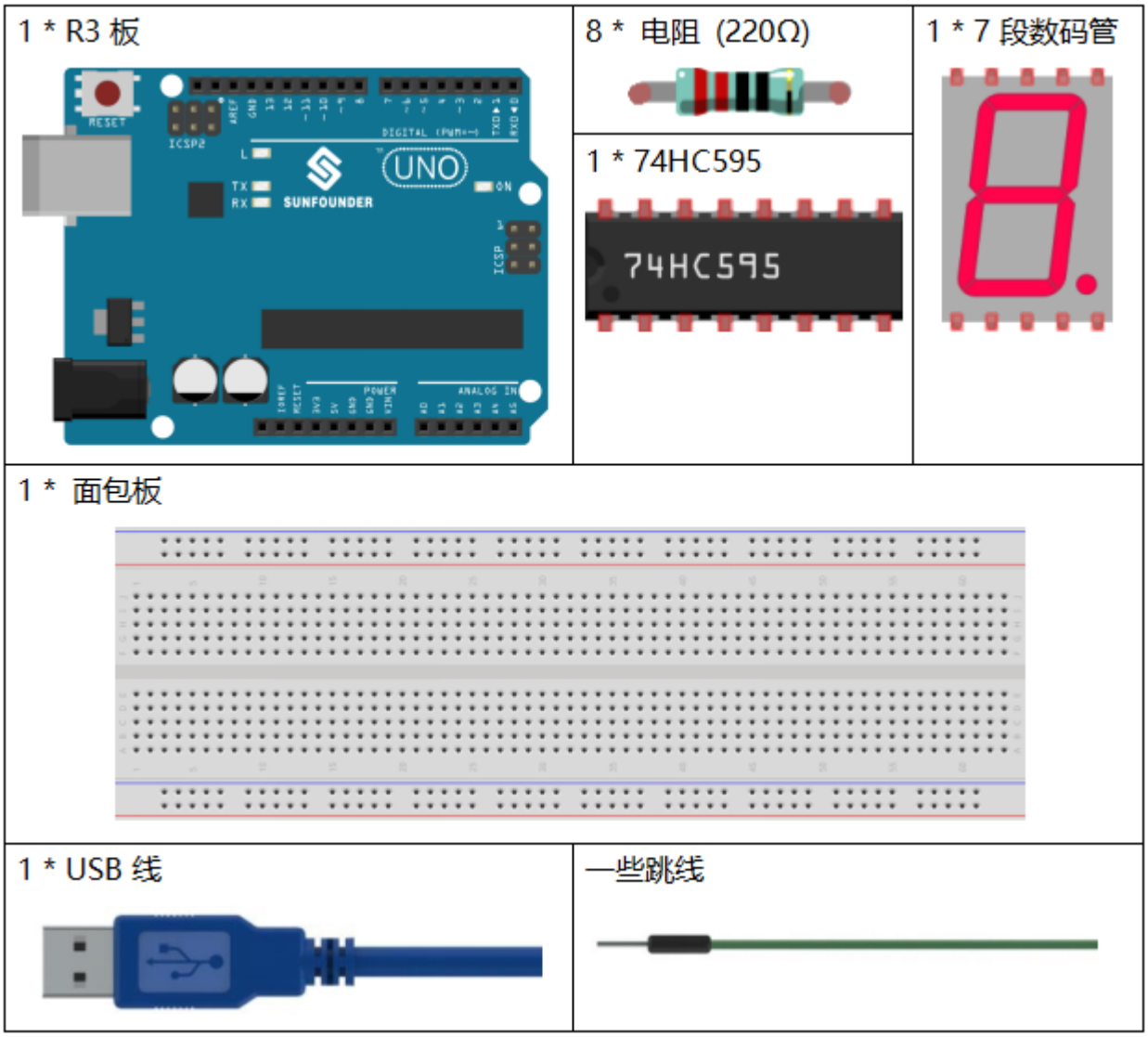
6.18 第 18 课 74HC595

6.18.1 介绍

通常，有两种方式可以驱动单个 7 段数码管。一种方法是将其 8 个引脚直接连接到主板上的 8 个端口，我们之前已经这样做了。或者你可以将 74HC595 连接到主板的三个端口，然后将 7 段数码管连接到 74HC595。

在本实验中，我们将使用后者。这样，我们可以节省五个端口——考虑到主板的端口有限，这一点非常重要。现在让我们开始吧！

6.18.2 所需器件



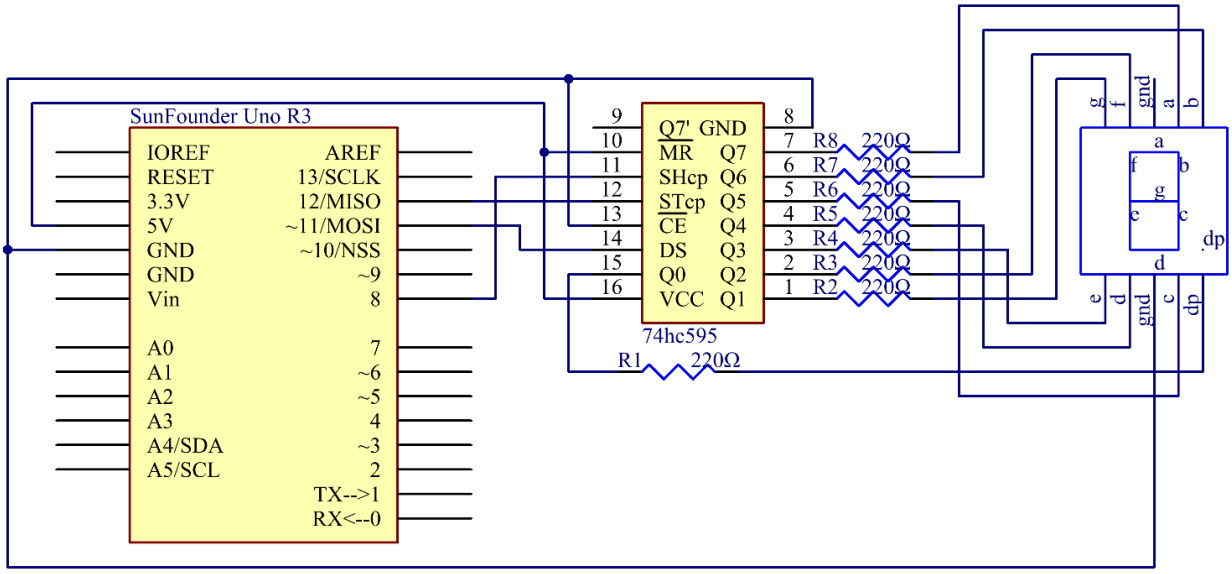
- *SunFounder R3 板*
- *面包板*
- *跳线*
- *电阻*
- *7 段数码管*
- *74HC595*

6.18.3 原理图

在实验中，MR（引脚 10）连接到 5V（高电平），OE（引脚 1）连接到 GND（低电平）。因此，数据在 SHcp 的上升沿输入，通过上升沿进入内存寄存器。我们使用 `shiftout()` 函数通过 DS 输出一个 8 位的数据到移位寄存器。在 SHcp 的上升沿，移位寄存器中的数据一次连续移动一位，即 Q1 中的数据移动到 Q2，以此类推。在 STcp 的上升沿，移位寄存器中的数据移动到内存寄存器中。

所有数据将在 8 次后移动到内存寄存器。然后将内存寄存器中的数据输出到总线（Q0-Q7）。所以 16 个字符依次显示在 7 段数码管中。

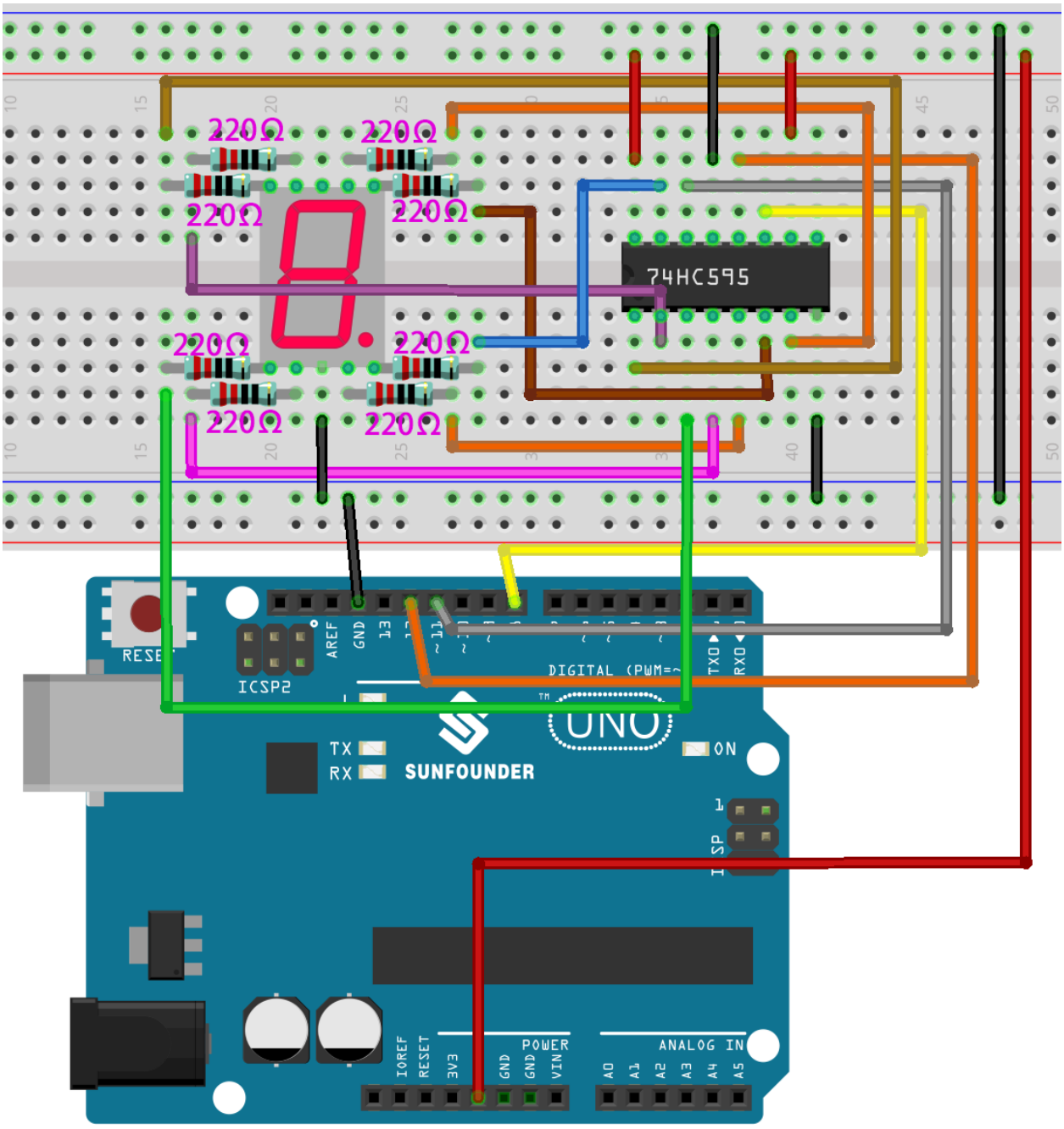
原理图如下所示：



6.18.4 实验步骤

第 1 步：搭建电路。(注意芯片的槽口方向，应该是在左侧)

7 段数码管	74HC595	R3 板
a	Q7	
b	Q6	
c	Q5	
d	Q4	
e	Q3	
f	Q2	
g	Q1	
DP	Q0 VCC DS CE ST SH MR Q7’ GND	5V 11 GND 12 8 5V N/C GND
•		GND

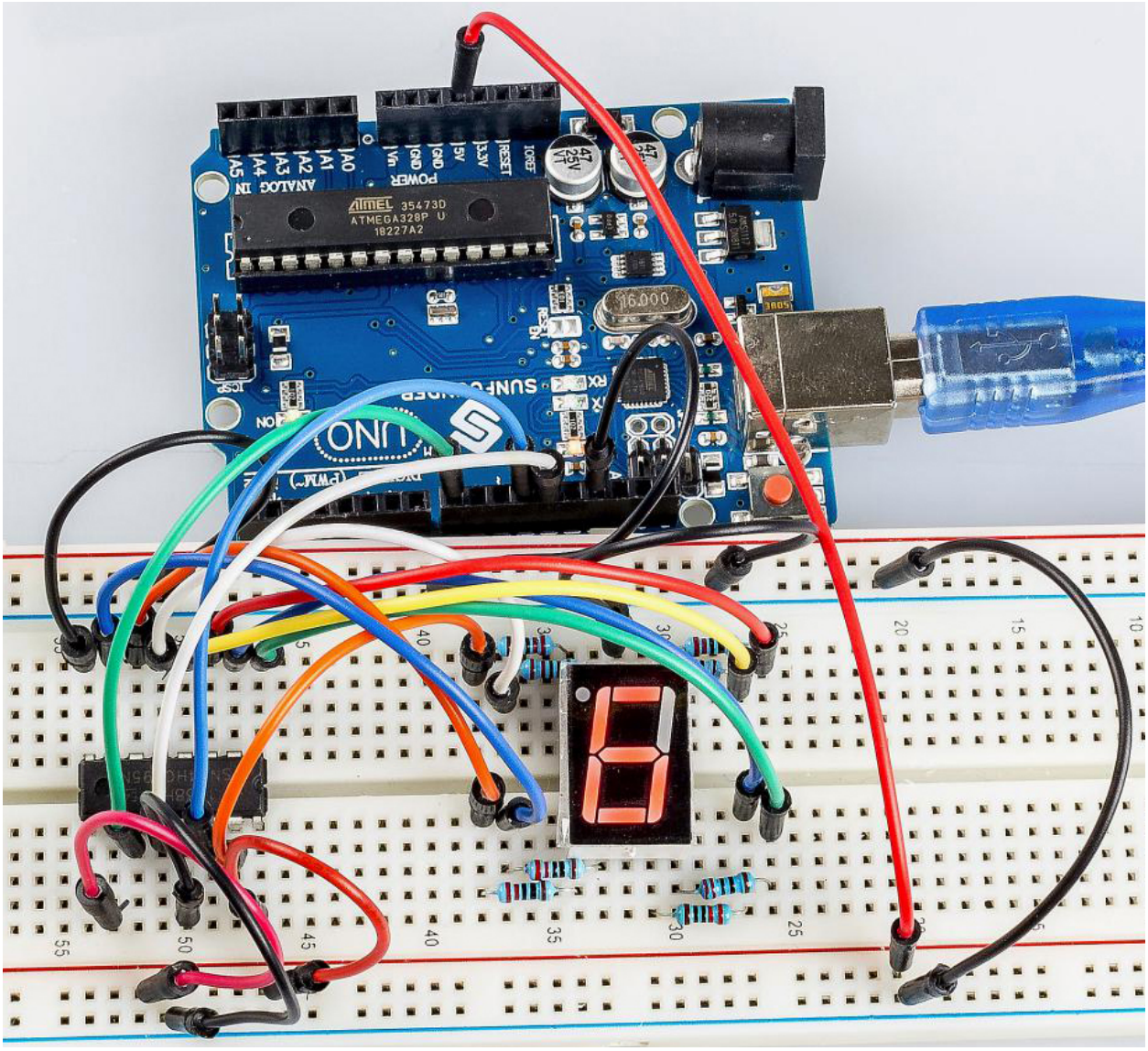


第 2 步：打开代码文件 `Lesson_18_74HC595.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

你将看到 7 段数码管显示 0-9，A-F。



6.18.5 代码

6.18.6 代码分析

设置数组和元素

```
int datArray[16] = {252, 96, 218, 242, 102, 182, 190, 224, 254, 246, 238, 62, 156,␣
↪122, 158, 142};
```

这个数组存放了从 0 到 F 的 16 个字符的数据，252 代表 0，可以自己计算。要显示 0，7 段数码管的 g 段（中间的）必须是低电平（暗）。

由于 g 连接到 74HC595 的 Q1，将 Q1 和 DP（点）都设置为低电平，其余引脚为高电平。因此，Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 的值为 1 1 1 1 1 1 0 0。

将二进制数改为十进制数：1x2^7 + 1x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 =252

这就是要显示的数字 0 的值。你可以类似地计算其他字符。

在 7 段数码管上显示 0-F

```
for(int num = 0; num < 16; num++)
{
    digitalWrite(STcp,LOW); //ground ST_CP and hold low for as long as you are␣
↪transmitting
    shiftOut(DS,SHcp,MSBFIRST,datArray[num]);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(STcp,HIGH); //pull the ST_CPST_CP to save the data
    delay(1000); //wait for a second
}
```

将 STcp 设置为低电平，然后设置为高电平。它将产生一个的上升沿脉冲。

shiftOut() 用于逐位移出一个字节的数据，即将 dataArray[num] 中的一个字节数据移到 DS 引脚的移位寄存器中。MSBFIRST 表示从高位移动。之后 digitalWrite（STCP，HIGH）运行时，STCP 将在上升沿。这时，移位寄存器中的数据就会被移到内存寄存器中。

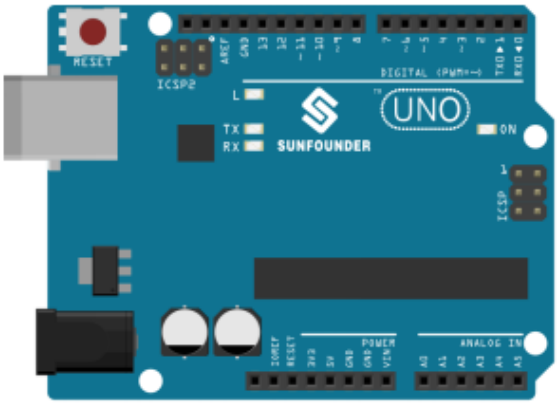
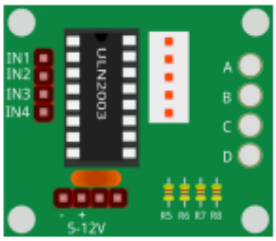
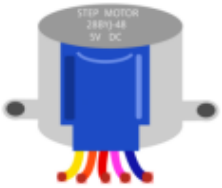
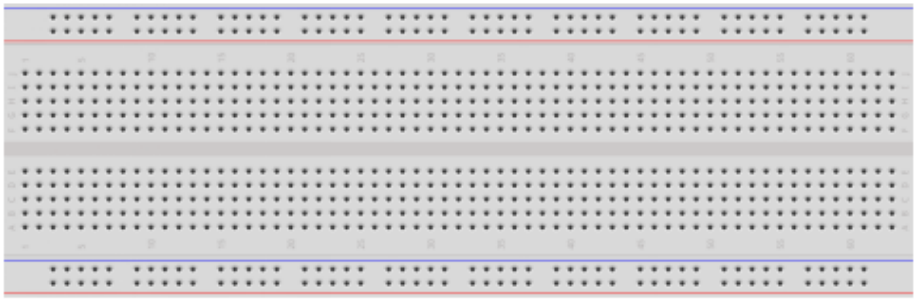
8 次后，一个字节的数据将被传送到内存寄存器中。然后将内存寄存器的数据输出到总线（Q0-Q7）。你将看到一个字符显示在 7 段数码管上，然后延迟 1000ms。在该行之后，返回 for()。如此循环直到 16 次后，7 段数码管中的所有字符都一一显示出来。

6.19 第 19 课步进电机

6.19.1 介绍

步进电机由于其独特的设计，可以在没有任何反馈机制的情况下进行高精度控制。步进电机的轴上装有一系列磁铁，由一系列电磁线圈控制，这些线圈按特定顺序带正负电，以小“步”精确地向前或向后移动。

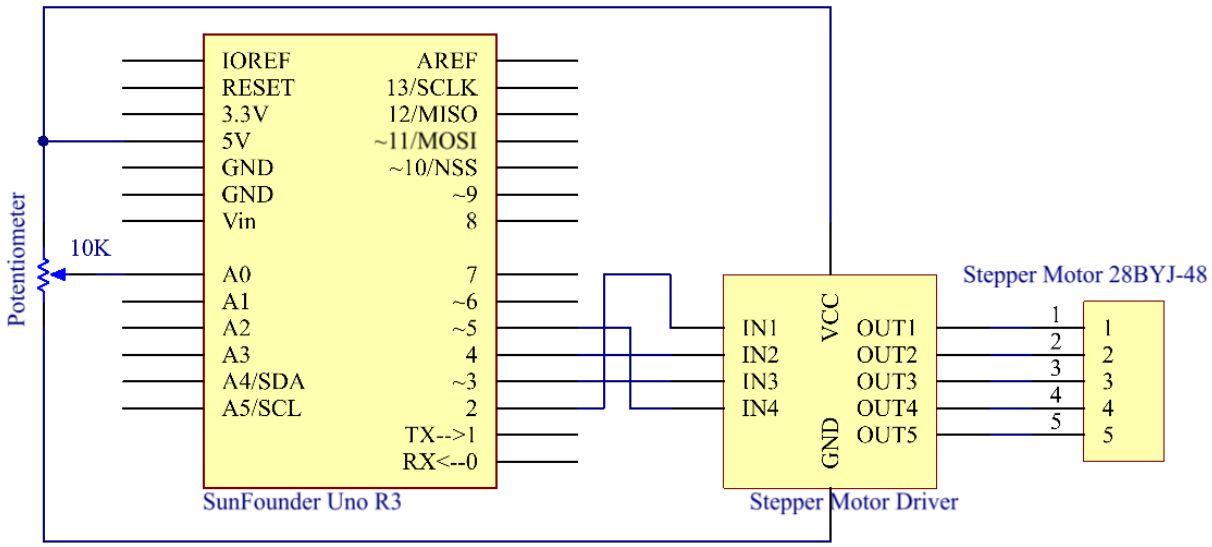
6.19.2 所需器件

<table>
<tr><td>1 * R3 板
  </td><td>1 * 步进电机驱动板
  </td><td>1 * 步进电机
 
1 * 电位器
  </td></tr>
<tr><td colspan="3">1 * 面包板
  </td></tr>
<tr><td>1 * USB 线
  </td><td colspan="2">一些跳线
  </td></tr>
</table>

- *SunFounder R3 板*
- 面包板
- 跳线
- 电位器
- 步进电机

6.19.3 原理图

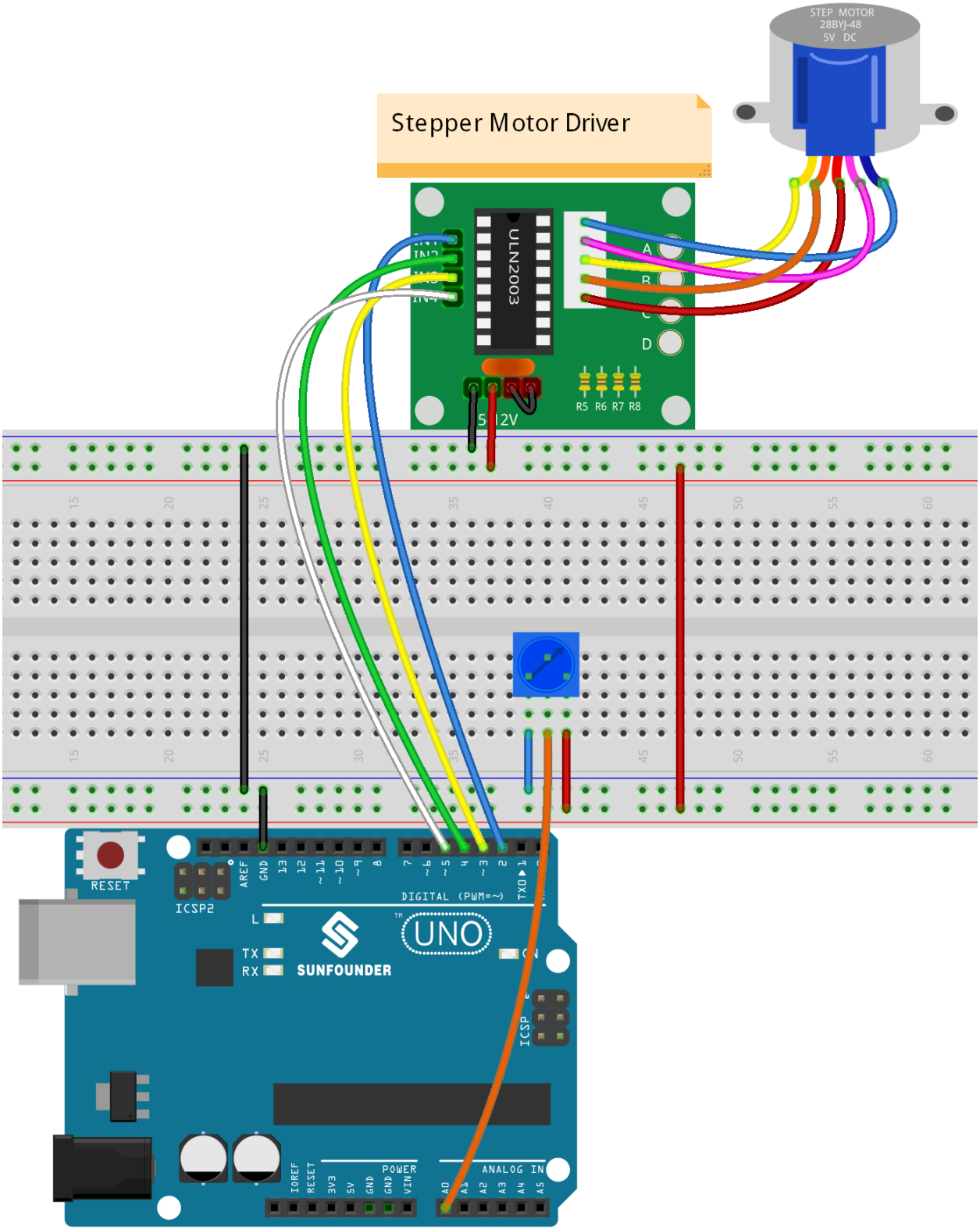
原理图如下所示：



6.19.4 实验步骤

第 1 步：搭建电路。

步进电机驱动板	R3 板
IN1	2
IN2	4
IN3	3
IN4	5
GND	GND
VCC	5v

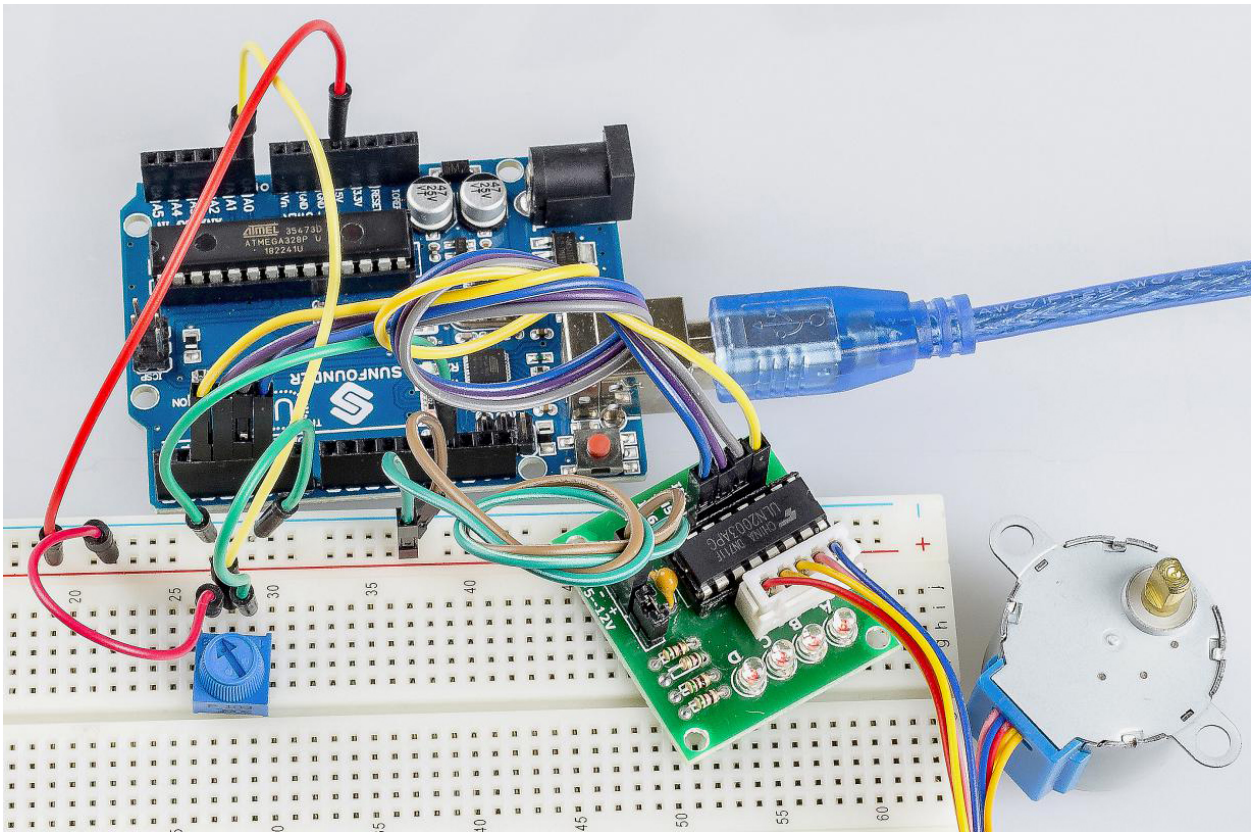


第 2 步：打开代码文件 `Lesson_19_Stepper_Motor.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你应该看到步进电机的摇臂顺时针和逆时针交替旋转。



6.19.5 代码

6.19.6 代码分析

初始化步进电机

```
#include <Stepper.h> //include a head file
//the steps of a circle
#define STEPS 2048
//set steps and the connection with MCU
Stepper stepper(STEPS, 2, 3, 4, 5);
//available to store previous value
int previous = 0;
```

包含头文件 `Stepper.h`，将步长设置为 100，然后使用函数 `stepper()` 初始化步进电机。

- `Stepper(steps, pin1, pin2, pin3, pin4)`：此函数创建 `Stepper` 类的新实例，代表连接到 Arduino 板的特定步进电机。
- `steps`：电机旋转一圈的步数。如果你的电机给出每步的度数，将该数字除以 360 以获得步数（例如 360 / 3.6 给出 100 步，整数型）。

setSpeed() 函数

```
//speed of per minute
stepper.setSpeed(15); //set the motor speed in rotations per minute(RPMs)
```

- `setSpeed(rpms)`：以每分钟转数 (RPMs) 为单位设置电机速度。此函数不会使电机转动，只是设置调用 `step()` 时的速度。
- `rpms`：电机每分钟旋转的速度 - 一个正数（长型）。

主程序

```
void loop()
{
//get analog value
int val = analogRead(0); //Read the value of the potentiometer
//current reading minus the reading of history
stepper.step(val - previous); //Turn the motor in val-previous steps
//store as prevous value
previous = val; //the value of potentiometer assignment to variable previous
}
```

主程序是先读取 A0 的值，然后根据 A0 的值来设置步进电机转动的步数。

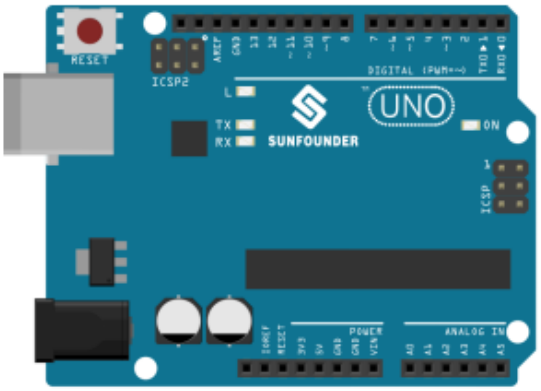
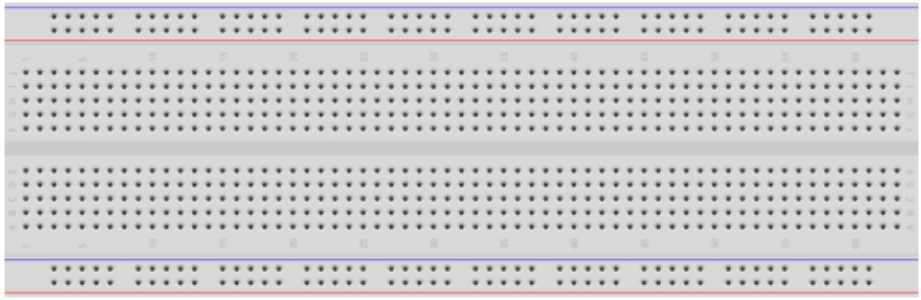
- `step(steps)`：以特定的步数转动电机，速度由最近调用 `setSpeed()` 确定。这个功能是阻塞的；也就是说，它将等到电机完成移动后才能将控制权传递给代码中的下一行。例如，如果你将速度设置为 1 RPM 并在 100 步电机上调用 `step(100)`，则此函数将需要整整一分钟才能运行。为了更好地控制，保持高速并且每次调用 `step()` 时只走几步。
- `steps`：转动电机的步数 - 正向转动一个方向，负向转动另一个（int）。

6.20 第 20 课简单创作 - 秒表

6.20.1 介绍

在这个课程中，我们将使用 4 位 7 段数码管来制作一个秒表。

6.20.2 所需器件

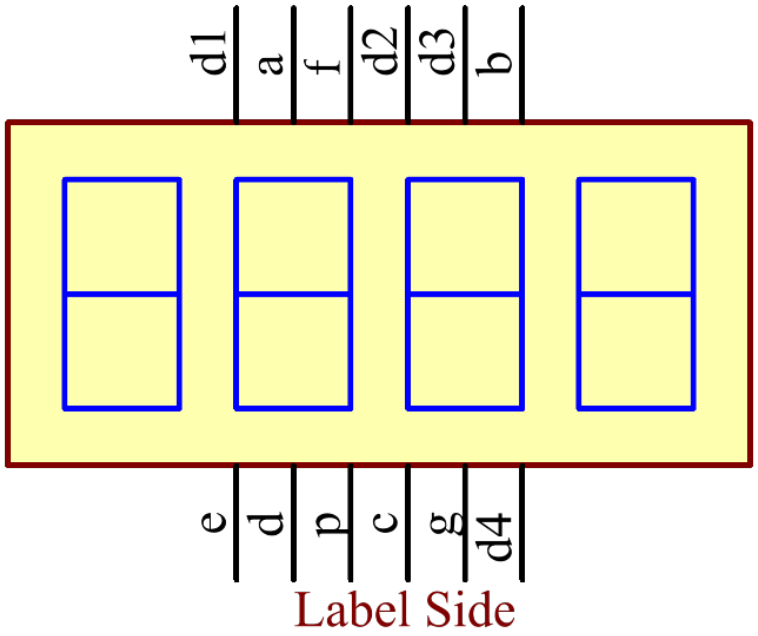
1 * R3 板 	8 * 电阻 (220Ω) 	1 * 4-Digit 7 段数码管 
1 * 面包板 		
1 * USB 线 	一些跳线 	

- *SunFounder R3 板*
- *面包板*
- *跳线*
- *电阻*
- *4 位 7 段数码管*

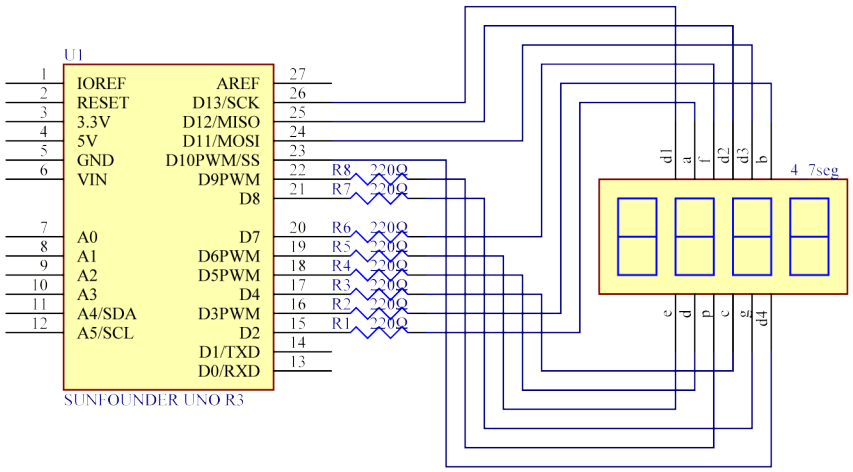
6.20.3 原理图

使用 7 段数码管时，如果是共阳极数码管，将阳极引脚接电源；如果是共阴极，则将阴极引脚连接到 GND。使用 4 位 7 段数码管时，共阳极或共阴极管脚控制显示的数字。只有一位数字有效，但是，根据视觉暂留原理，我们可以看到四个 7 段数码管都显示数字。这是因为电子扫描速度太快，我们无法注意到间隔。

4 位 7 段数码管示意图如下：



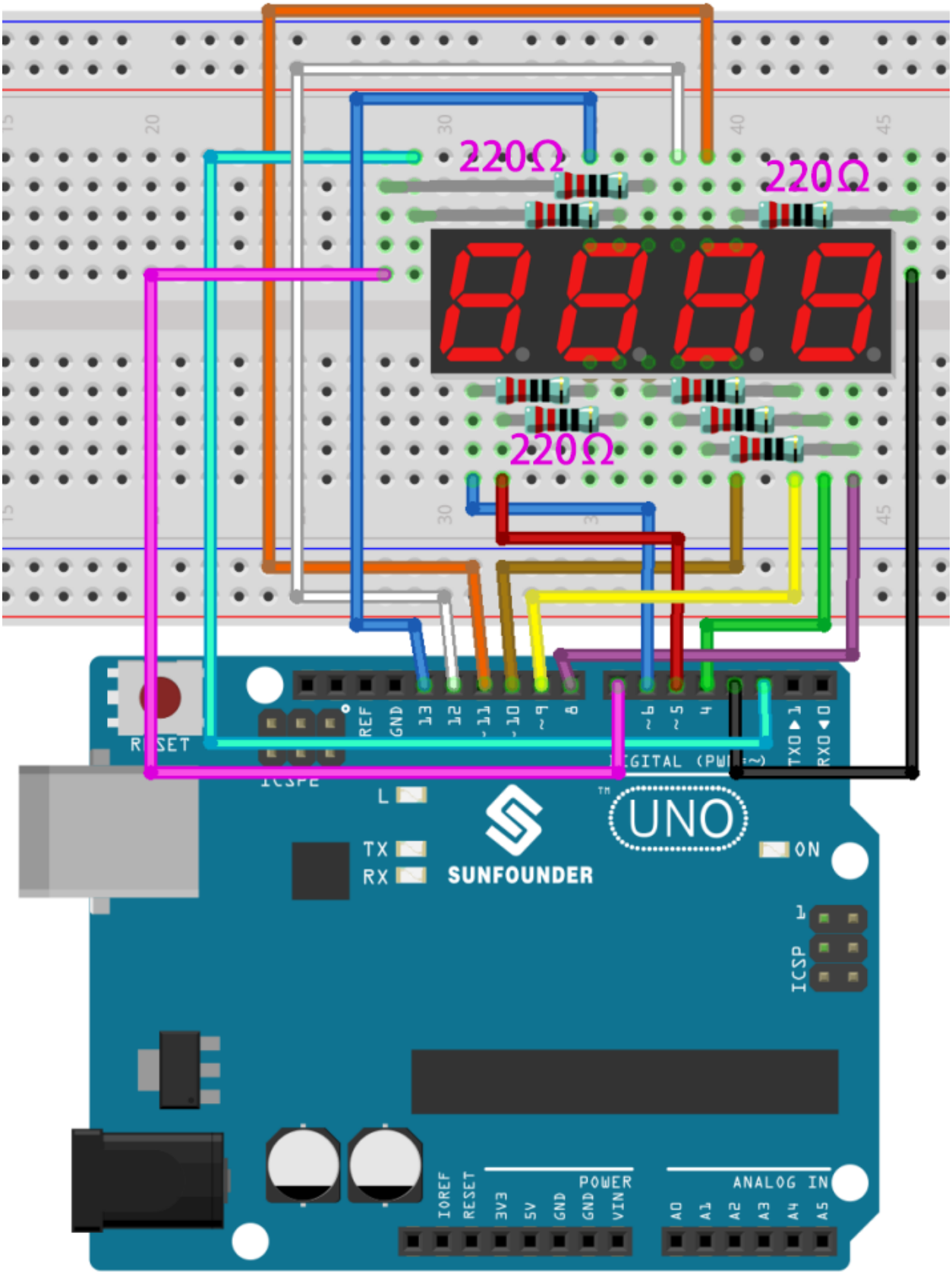
原理图如下所示：



6.20.4 实验步骤

第 1 步：搭建电路。.

4 位 7 段数码管	R3 板
a	2
b	3
c	4
d	5
e	6
f	7
g	8
p	9
D1	13
D2	12
D3	11
D4	10

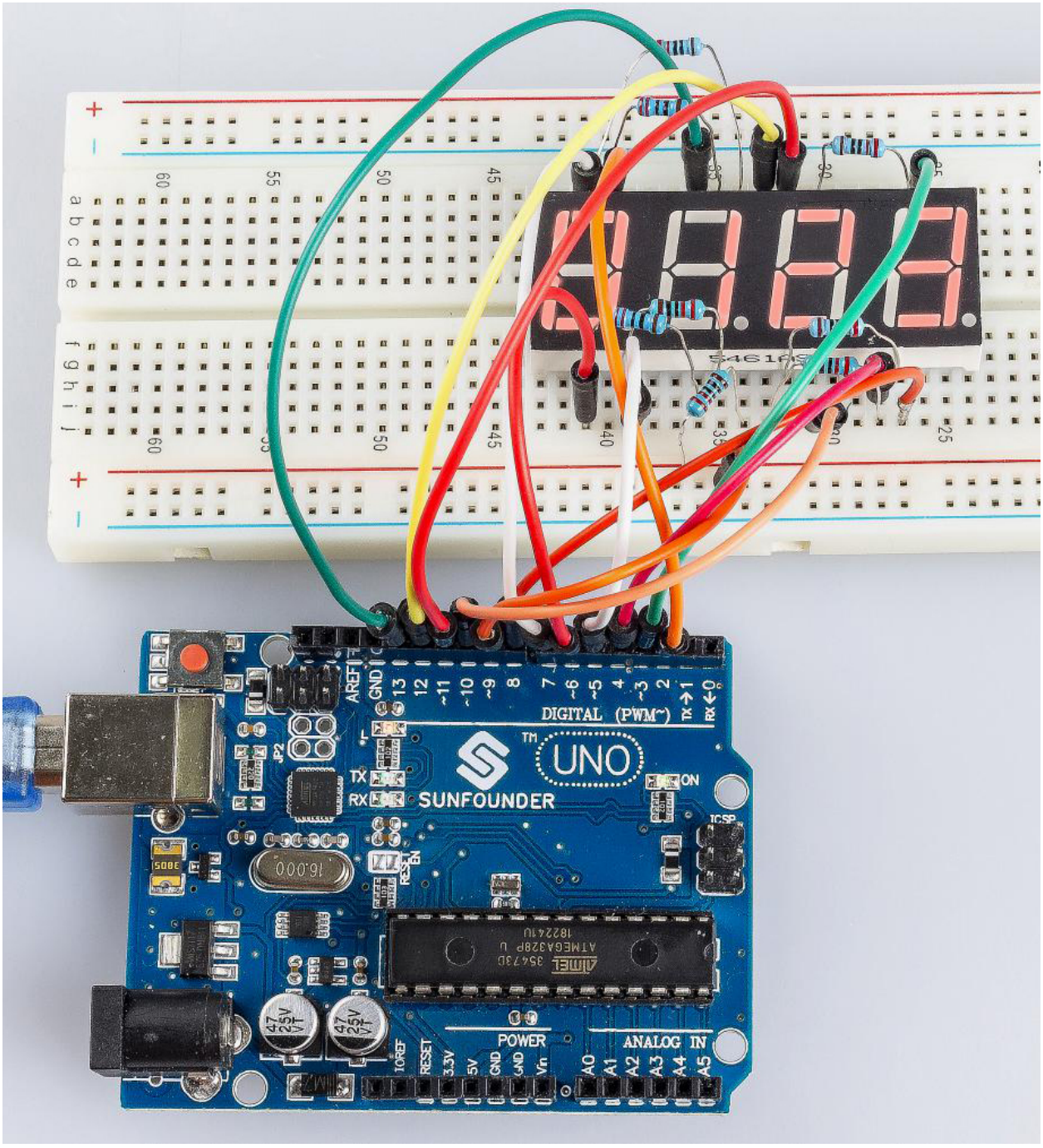

220Ω
220Ω
220Ω
UNO
SUNFOUNDER
POWER
ANALOG IN

第 2 步：打开代码文件 `Lesson_20_Stopwatch.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你可以在 4 位 7 段数码管上看到数字每秒增加 1。



6.20.5 代码

6.20.6 代码分析

这就是代码的全部内容，比较长，我总结一下：

将 4 位 7 段数码管的所有引脚设置为输出。设置定时器 1 为 0.1 秒，所以当 0.1 秒的时候，`add()` 会被调用；但是在 0.1 秒过去之前，`add()` 还没有被调用。然后运行 `loop()` 函数，4 个数码管显示为 0000。等待一段时间，0.1 秒后，表明 count=10，调用函数 `add()`。然后 n++=1；因为 1<10000，不会恢复到 0。运行 `loop()`，数码管显示为 0001。0.1 秒后，n 增加 1，n++=2，显示将变成 0002，然后是 0003，一直到 9999。n 每秒增加 1，显示的数字也相应增加，直到 n=10000，n 再次为 0。然后从 0 开始计数。

初始化定时器

```
Timer1.initialize(100000);
// set a timer of length 100000 microseconds(or 0.1 sec - or 10Hz => the led will␣
↪blink 5 times, 5 cycles of on-and-off, per second)

Timer1.attachInterrupt( add ); // attach the service routine here
```

语句 `attachInterrupt(add)` 就是附加一个 ISR 函数，当有中断时调用 `add()` 函数。

Loop 函数

```
void loop()
{
    clearLEDs();//clear the 7-segment display screen
    pickDigit(0);//Light up 7-segment display d1
    pickNumber((n / 1000)); // get the value of thousand
    delay(del);//delay 5ms

    clearLEDs();//clear the  7-segment display screen
    pickDigit(1);//Light up 7-segment display d2
    pickNumber((n % 1000) / 100); // get the value of hundred
    delay(del);//delay 5ms

    clearLEDs();//clear the  7-segment display screen
    pickDigit(2);//Light up 7-segment display d3
    pickNumber(n % 100 / 10); //get the value of ten
    delay(del);//delay 5ms

    clearLEDs();//clear the 7-segment display screen
    pickDigit(3);//Light up 7-segment display d4
    pickNumber(n % 10); //Get the value of single digit
    delay(del);//delay 5ms
}
```

`loop()` 用于让四段显示器显示一个数值的个位数、十位、十万位。

如 n=1345、(1345/1000)=1、(1345%1000)/100)=3、((1345%100)/10)=4、(n%10)=5

pickDigit(int x) 函数

```
void pickDigit(int x)  //light up a 7-segment display
{
    //The 7-segment LED display is a common-cathode one. So also use digitalWrite to␣
↪set d1 as high and the LED will go out
    digitalWrite(d1, HIGH);
    digitalWrite(d2, HIGH);
```

(续下页)

(接上页)

```
    digitalWrite(d3, HIGH);
    digitalWrite(d4, HIGH);

    switch (x)
    {
        case 0:
            digitalWrite(d1, LOW);//Light d1 up
            break;
        case 1:
            digitalWrite(d2, LOW); //Light d2 up
            break;
        case 2:
            digitalWrite(d3, LOW); //Light d3 up
            break;
        default:
            digitalWrite(d4, LOW); //Light d4 up
            break;
    }
}
```

4 位 7 段数码管为共阴的，将 d1、d2、d3、d4 全部设置为 HIGH 使其熄灭。

再来判断 x 的值：

- x 为 0，让 d1 为低电平来让第 4 个数码管（左边第一个）工作。
- x 为 1，让第 3 个数码管工作。
- x 为 2，让第 2 个数码管工作。
- 默认情况下，让第 1 个数码管（右边第一个）工作。

pickNumber(int x) 函数

```
void pickNumber(int x)
{
    switch (x)
    {
        default:
            zero();
            break;
        case 1:
            one();
            break;
        case 2:
            two();
            break;
        case 3:
            three();
            break;
        ...
```

这个函数的功能是控制 LED 显示数字。调用 `zero()`、`one()` 直到 `nine()` 函数显示 0-9 数字。

通过 x 的值来判断显示什么数字：

- 默认情况，调用 `zero()` 函数来显示 0。
- x 为 1，调用 `one()` 函数来显示 1。
- x 为 2，调用 `two()` 函数来显示 2。

- x 为 3，调用 `three()` 函数来显示 3。
- x 为 4，调用 `four()` 函数来显示 4。
- x 为 5，调用 `five()` 函数来显示 5。
- x 为 6，调用 `six()` 函数来显示 6。
- x 为 7，调用 `seven()` 函数来显示 7。
- x 为 8，调用 `eight()` 函数来显示 8。
- x 为 9，调用 `nine()` 函数来显示 9。

以 `zero()` 为例：

`zero()` 函数是控制 LED 的高低电平。使用 `digitalWrite()` 将 a 设置为 f 为高，g 为低。根据刚才提到的引脚图，当 a 到 f 为高，g 为低时，会显示数字 0。

```
void zero() //the  7-segment led display 0
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, LOW);
}
```

clearLEDs() 函数

```
void clearLEDs() //clear the  7-segment display screen
{
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(p, LOW);
}
```

将 a-p 引脚都设置为低电平来让 4 位 7 段数码管全部熄灭。

add() 函数

```
void add()
{
    // Toggle LED
    count ++;
    if(count == 10)
    {
    count = 0;
    n ++;
    if(n == 10000)
    {
        n = 0;
    }
    }
}
```

`count` 的初始值是 0，将 `count` 累加；加到 10 再重置为 0，此时将 `n` 累加；`n` 加到 10000 后，再重置为 0。

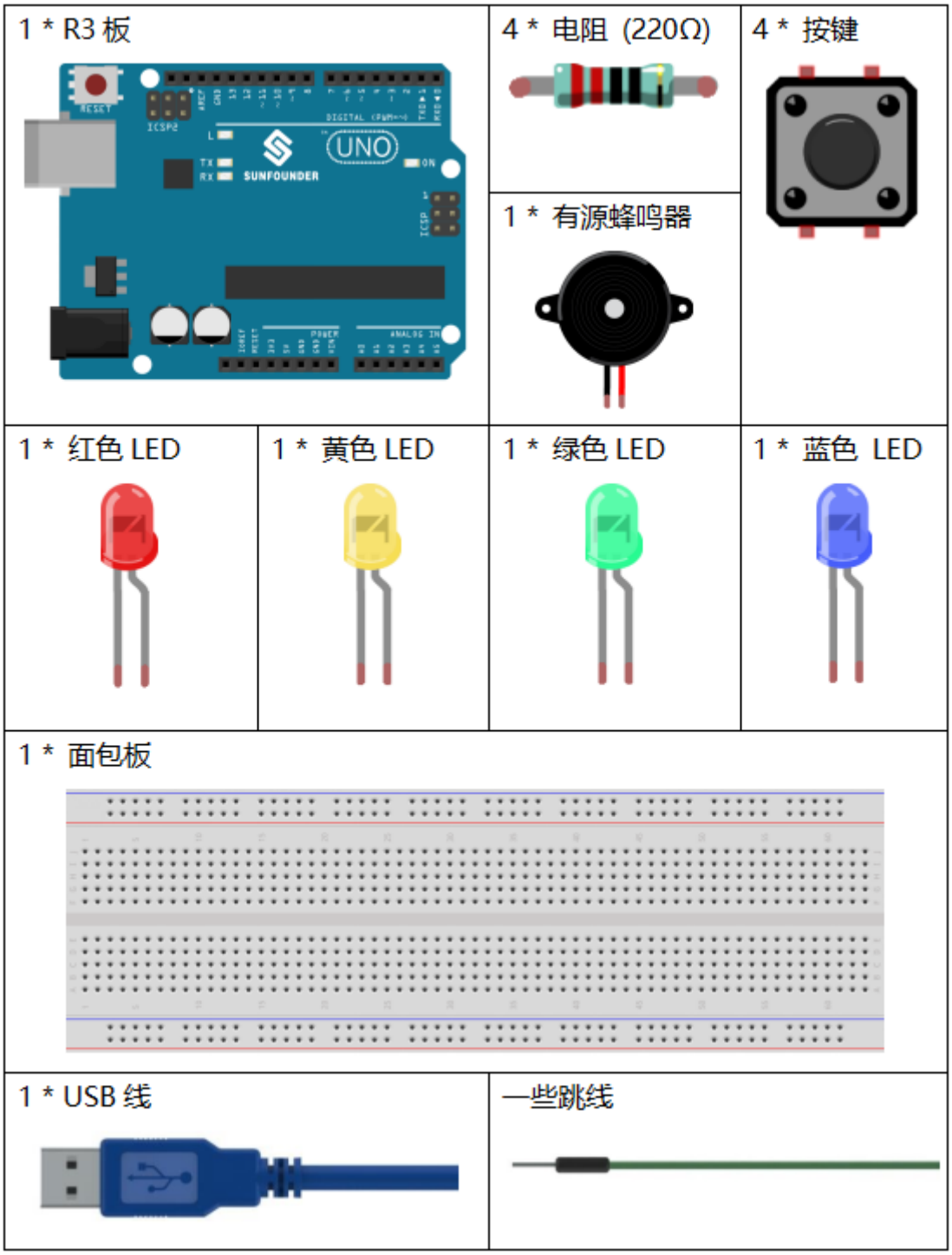
6.21 第 21 课简单创作 - 抢答器

6.21.1 介绍

在问答比赛中，尤其是娱乐活动（如竞技答题活动）中，主办方为了准确、公正、直观地确定答题者的座位号，往往会采用蜂鸣器系统。

现在系统可以用数据来说明判断的准确性和公平性，提高了娱乐性。同时，也更加公平公正。在本课中，我们将使用一些按键、蜂鸣器和 LED 来制作测验蜂鸣器系统。

6.21.2 所需器件



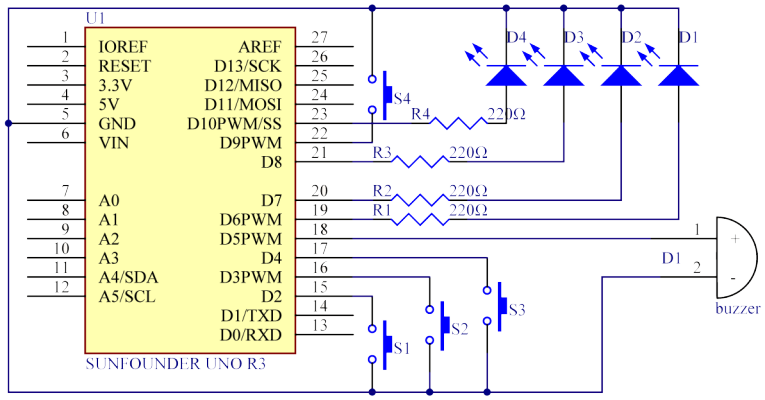
- *SunFounder R3 板*
- 面包板
- 跳线
- 电阻

- *LED 发光二极管*
- 按键
- 蜂鸣器

6.21.3 原理图

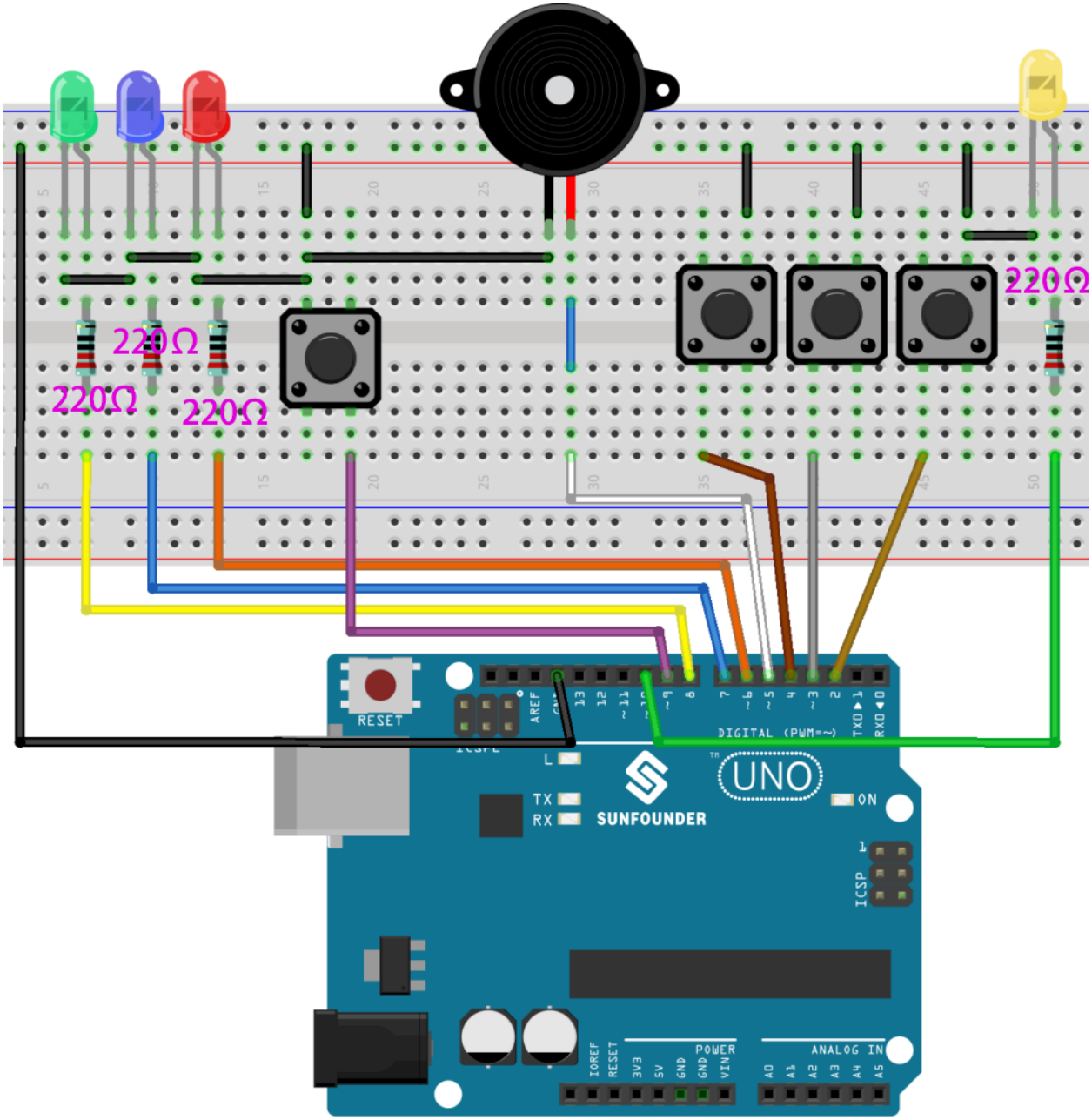
按键 1、2 和 3 是抢答按键，按键 4 是重置按键。如果先按下按键 1，蜂鸣器将发出蜂鸣声，相应的 LED 将亮起，所有其他 LED 将熄灭。如果要开始另一轮，请按按键 4 重置。

原理图如下所示：



6.21.4 实验步骤

第 1 步：搭建电路。

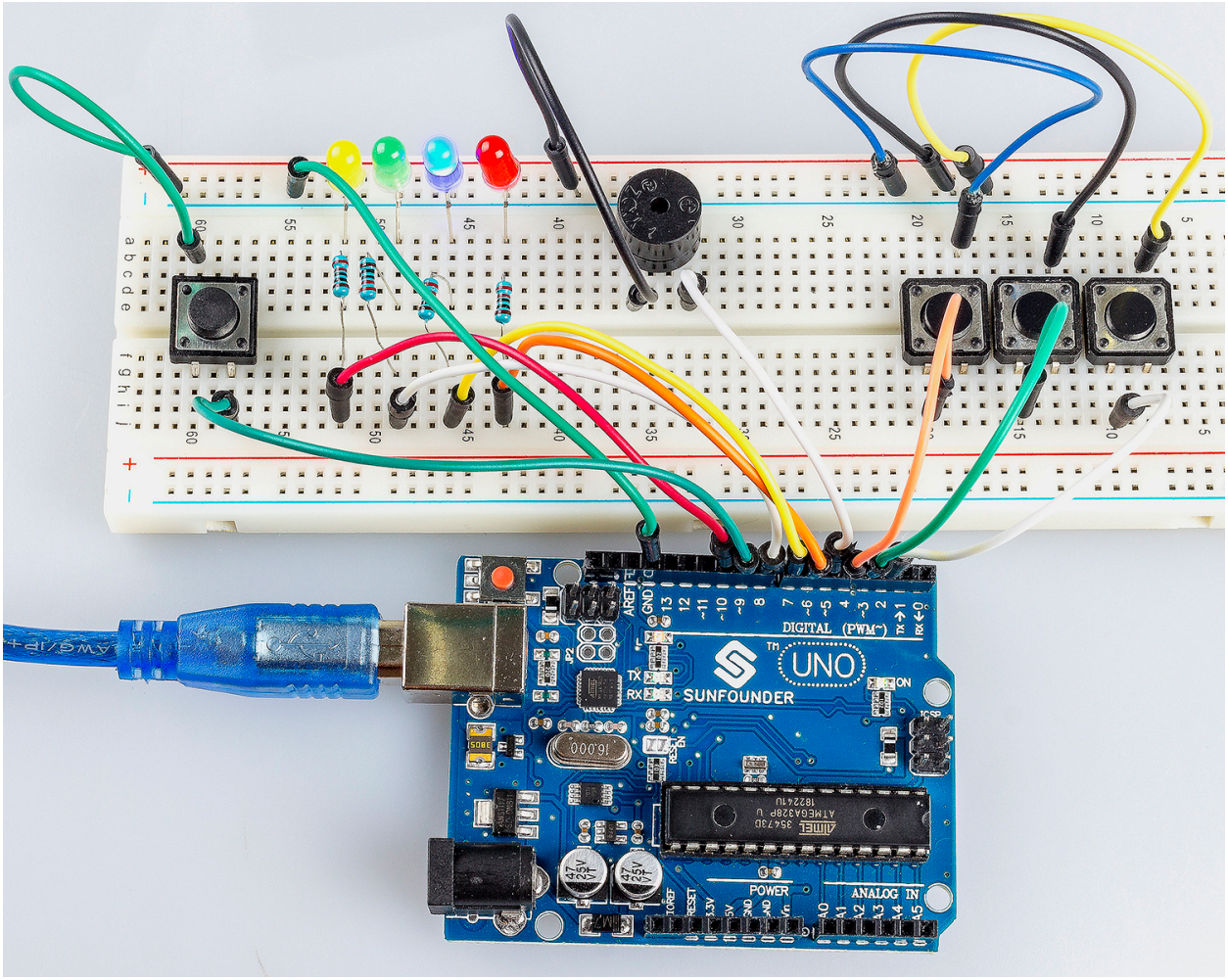


第 2 步：打开代码文件 `Lesson_21_Answer_Machine.ino`。

Step 3: Select the Board and Port.

第 4 步：点击 **上传**按钮来上传代码。

现在，首先按下按键 4 开始。如果你先按下按键 1，你将看到相应的 LED 亮起，蜂鸣器将发出哔哔声。然后再次按下按键 4 进行重置，然后再按下其他按键。



6.21.5 代码

6.21.6 代码分析

这个实验的代码可能有点长。但是语法很简单。

这个代码用到了 6 个嵌套 if 语句。

- 第一个 if 语句用来判断按键 4 是否按下。
- 第二个 if 语句用来再次判断按键 4 是否按下，用来防止误触。若确定按下，则让 flag 为 1，同时让 LED 点亮。
- 第三个 if 语句用来判断 flag 的值，如果为 1(按键 4 已按下)，此时读取按键 1，2，3 的值。
- 第四-六个 if 语句用来分别判断按键 1，2，3 是否按键，如果按下，则让 LED 点亮，蜂鸣器出声音。

Alarm() 函数

```
void Alarm()
{
for(int i=0;i<100;i++){
    digitalWrite(buzzerPin,HIGH); //the buzzer sound
```

(续下页)

(接上页)

```
    delay(2);
    digitalWrite(buzzerPin,LOW);  //without sound
    delay(2);                     //when delay time changed,the frequency changed
}
}
```

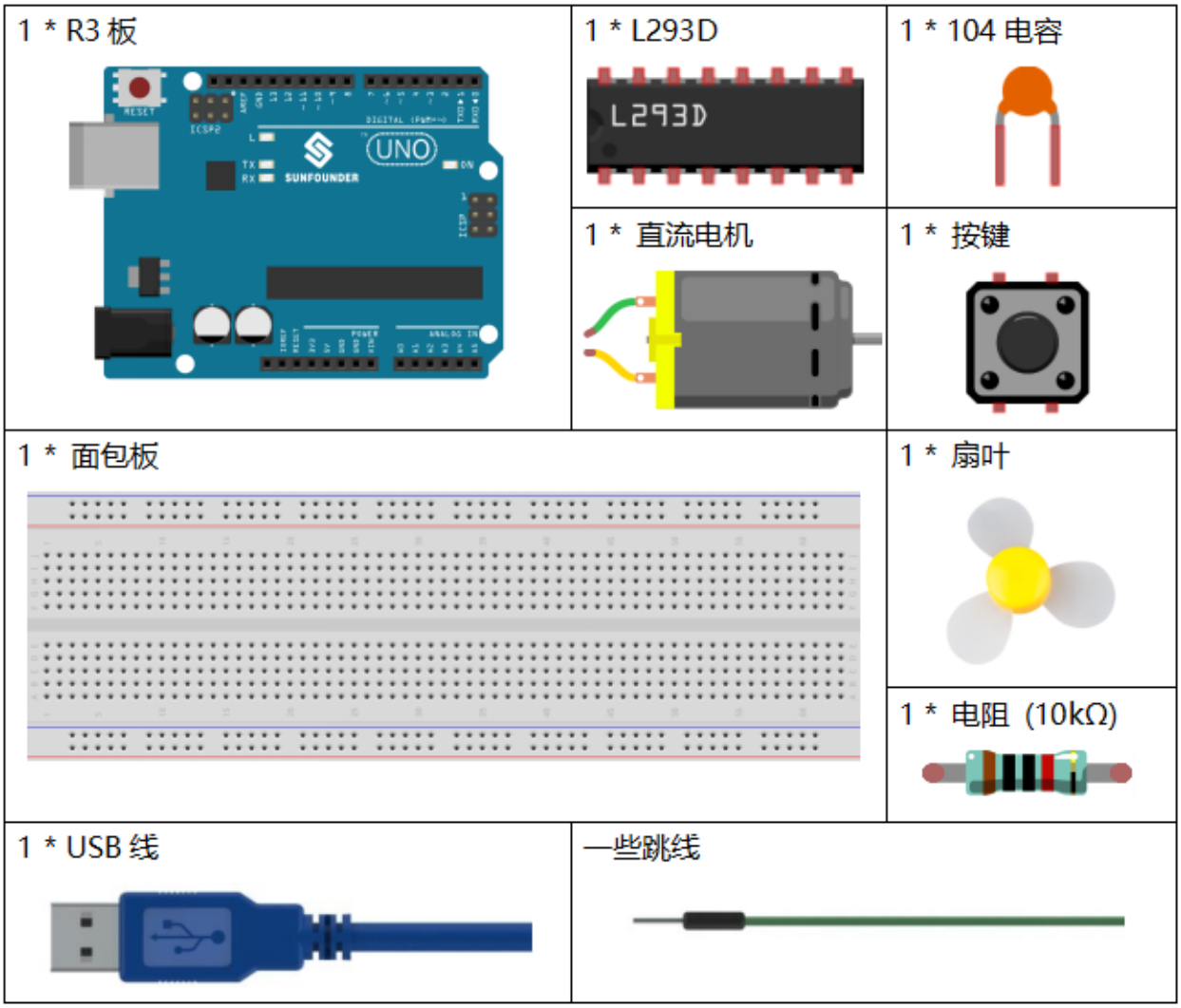
这个函数是用来设置蜂鸣器发出的声音长度和频率。

6.22 第 22 课简单创作 - 小风扇

6.22.1 介绍

在夏天若有个手持小风扇，让你能够走到哪吹到哪，并且还能更改档速，将是个非常爽的事情。
在这个课程中，我们将制作一个手持风扇原型 (去外壳)。

6.22.2 所需器件



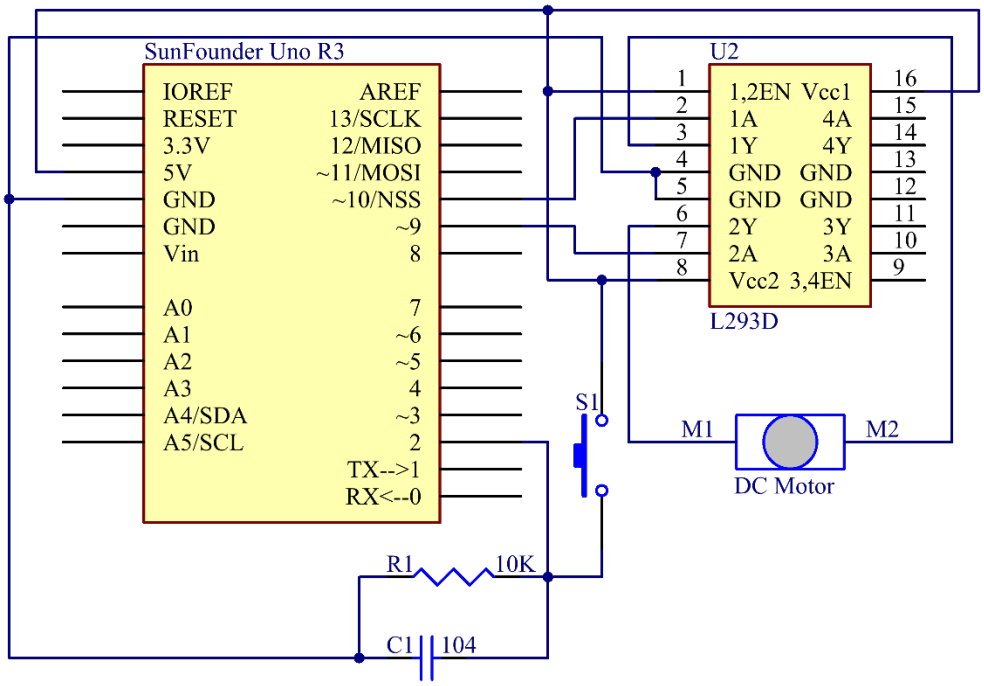
- *SunFounder R3 板*
- 面包板
- 跳线
- 电阻
- 电容
- 按键
- *L293D*
- 直流电机

6.22.3 原理图

Arduino I/O 端口的最大电流为 20mA，但电机的驱动电流至少为 70mA。因此，我们不能直接使用 I/O 口来驱动电流；相反，我们可以使用 L293D 来驱动电机。

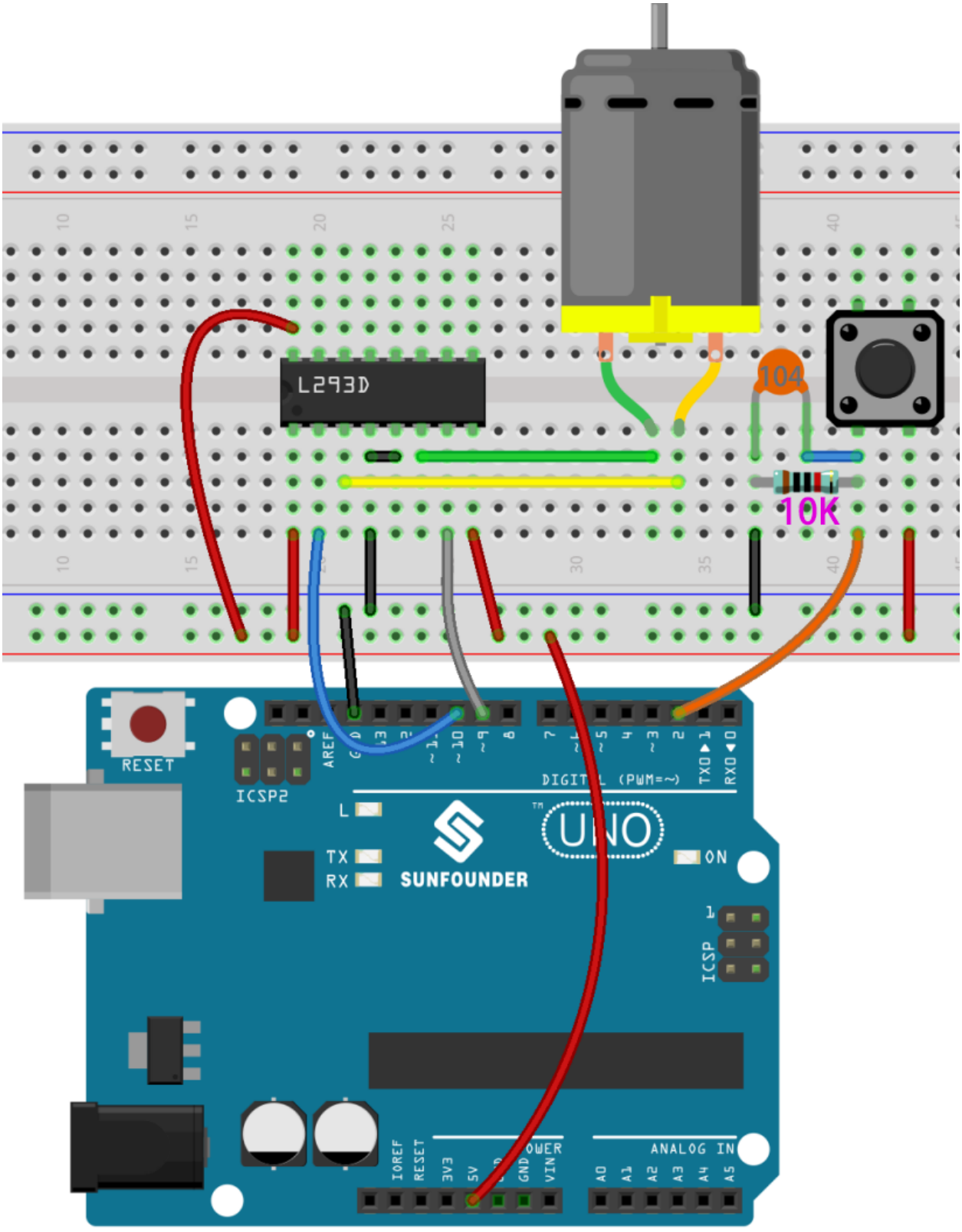
L293D 的 Enable pin 1,2EN 已经连接到 5V，所以 L293D 一直处于工作状态。将引脚 1A 和 2A 分别连接到控制板的引脚 9 和 10。电机的两个引脚分别连接到引脚 1Y 和 2Y。当 10 脚为高电平，9 脚为低电平时，电机开始向一个方向旋转。当引脚 10 为低电平且引脚 9 为高电平时，它以相反的方向旋转。

原理图如下所示：



6.22.4 实验步骤

第 1 步：搭建电路。

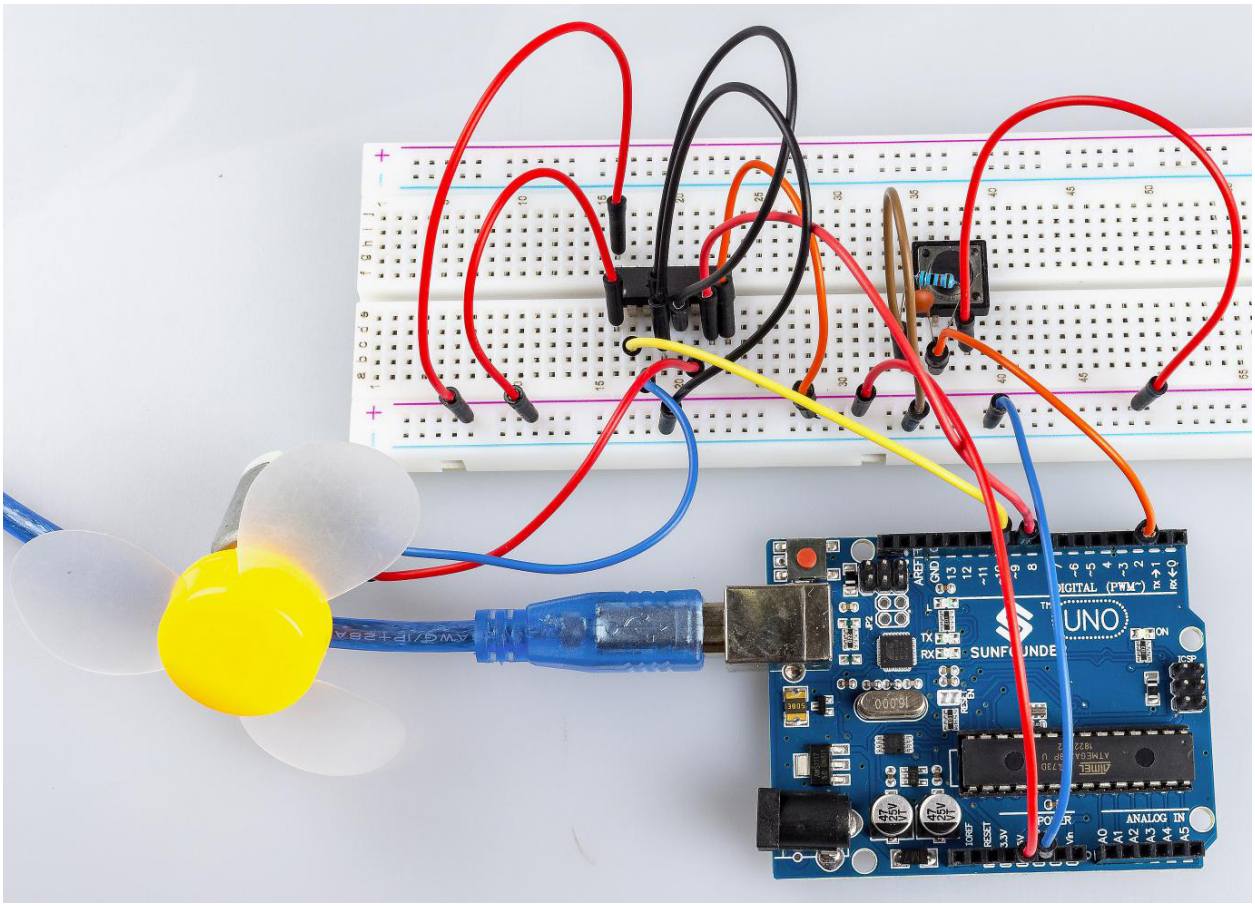


第 2 步：打开代码文件 `Lesson_22_Small_Fan.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

制作的小风扇有 3 档速度，按下第一次，风扇转速较慢，再次按下，风扇加速；第三次按下，风扇达到最大速度，按下第四次风扇将停止；以此为一个循环。



6.22.5 代码

6.22.6 代码分析

这个代码嵌套了 5 个 `if` 语句用来判断按键的按下状态。

- 第一个 if 语句用来判断按键是否按下。
- 第二个 if 语句用来判断时间是否过了 50ms。
- 第三个 if 语句用来判断过了 50ms, 按键确实有按下，以免有误触。
- 第四个 if 语句用来记录按键按下次数，每按下一次，`stat` 加 1。
- 第五个 if 语句用来判断按键按下次数是否大于 4，如果是，则将 `stat` 清零。

switch() 语句

```
switch(stat)
{
case 1:
    clockwise(rank1);// When stat=1, set the rotate speed of the motor as rank1=150
```

(续下页)

(接上页)

```
    break;
case 2:
    clockwise(rank2);// When stat=2, set the rotate speed of the motor as rank1=200
    break;
case 3:
    clockwise(rank3);// When stat=3, set the rotate speed of the motor as rank1=250
    break;
default:
    clockwise(0);
}
```

`switch` 语句与 `if` 语句一样，`switch case` 允许程序员在各种条件下执行的不同代码来控制程序流程。特别是，switch 语句将变量的值与 case 语句中指定的值进行比较。当找到值与变量的值匹配的 case 语句时，将运行该 case 语句中的代码。如果没有 `break` 语句，`switch` 语句将继续执行下面的表达式，直到 `break` 或到达 switch 语句的末尾。

在这部分代码中：

- 如果 stat = 1, 让风扇以速度 rank1(150) 转动。
- 如果 stat = 1, 让风扇以速度 rank2(200) 转动。
- 如果 stat = 1, 让风扇以速度 rank3(250) 转动。
- 如果 stat = 0, 让风扇以速度 0 转动。

clockwise() 函数

```
void clockwise(int Speed)//
{
    analogWrite(motorIn1,0);
    analogWrite(motorIn2,Speed);
}
```

该功能是设置电机的转速：将 `Speed` 写入引脚 9，将 0 写入引脚 10。电机朝某个方向旋转，速度为 `Speed` 的值。

6.22.7 实验总结

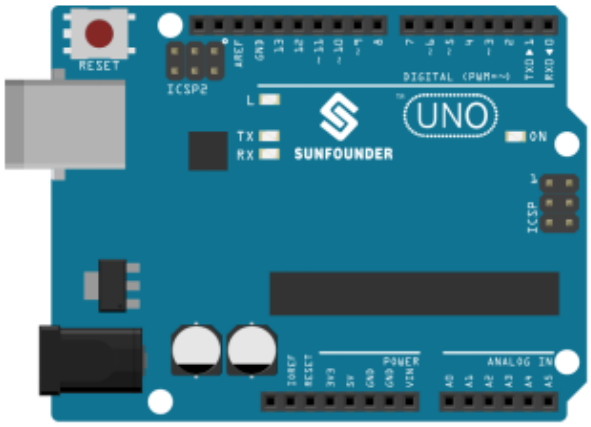
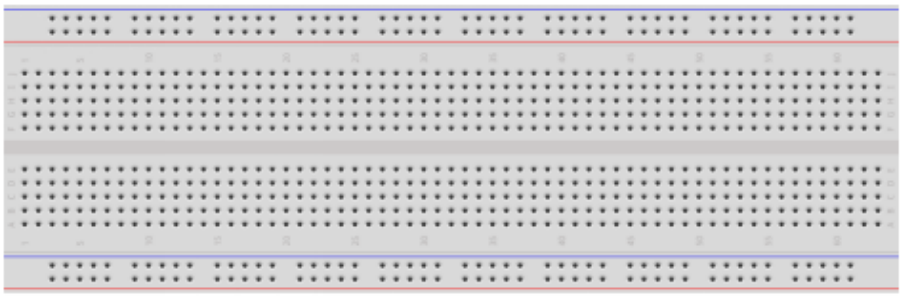
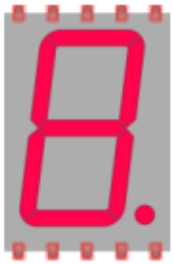
在本实验中，你还可以控制电机转动与否。只需将 L293D 的引脚 1、2EN 连接到控制板的 I/O 端口。设置 1、2EN 为高电平，电机开始转动；将其设置为低电平，它将停止旋转。

6.23 第 23 课简单创作 - 数字骰子

6.23.1 介绍

在之前的实验中，我们学习了如何使用 7 段数码管并通过按键控制 LED。在本课中，我们将使用一个 7 段数码管和一个按键来创建一个简单的数字骰子。

6.23.2 所需器件

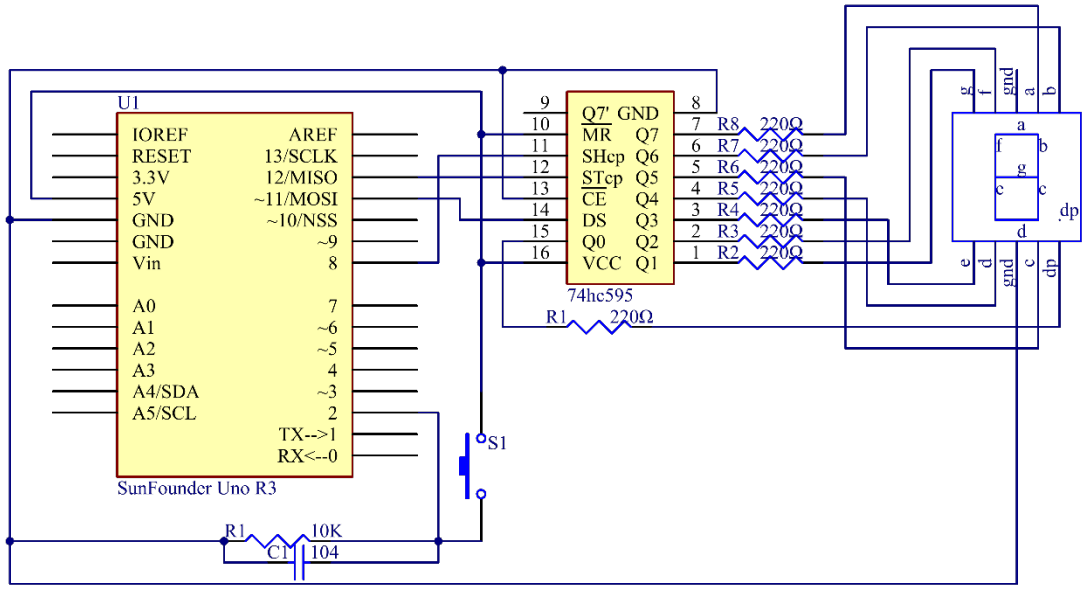
<table>
<tr><td>1 * R3 板
  </td><td>1 * 104 电容
  </td><td>1 * 按键
  </td></tr>
<tr><td rowspan="2">1 * 面包板
  </td><td>8 * 电阻 (220Ω)
  </td><td rowspan="2">1 * 74HC595
  </td></tr>
<tr><td>1 * 电阻 (10kΩ)
  </td></tr>
<tr><td colspan="2">1 * 7 段数码管
  </td></tr>
<tr><td>1 * USB 线
  </td><td colspan="2">一些跳线
  </td></tr>
</table>

- *SunFounder R3 板*
- *面包板*
- *跳线*
- *电阻*
- *7 段数码管*
- *74HC595*
- *按键*
- *电容*

6.23.3 原理图

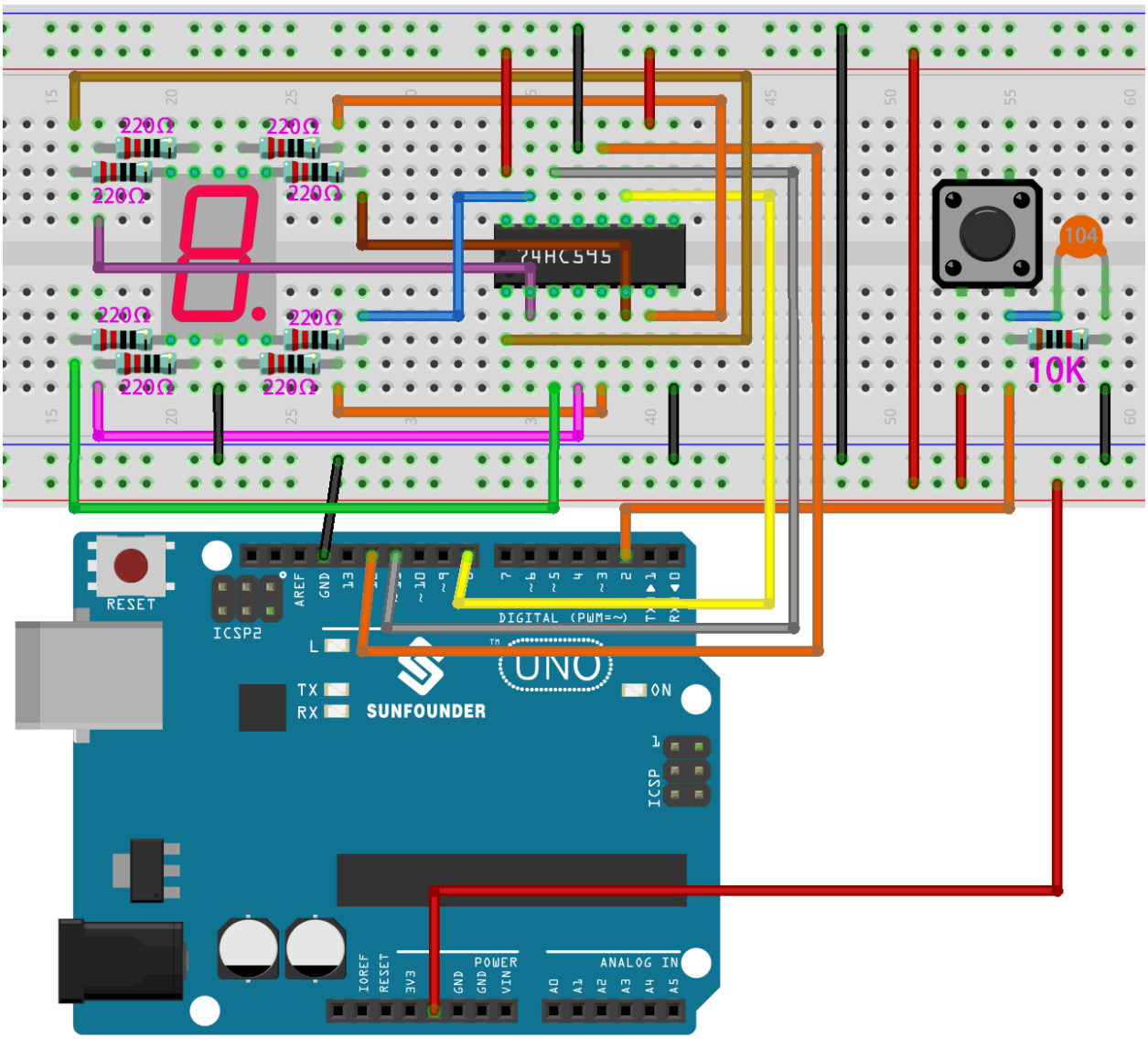
数字骰子背后的想法非常简单：一个 7 段数码管从 1 到 7 快速循环显示。当按下按键时，流动会减慢，直到它停在一个数字上。当再次按下按键时，该过程将重复。

原理图如下所示：



6.23.4 实验步骤

第 1 步：搭建电路.

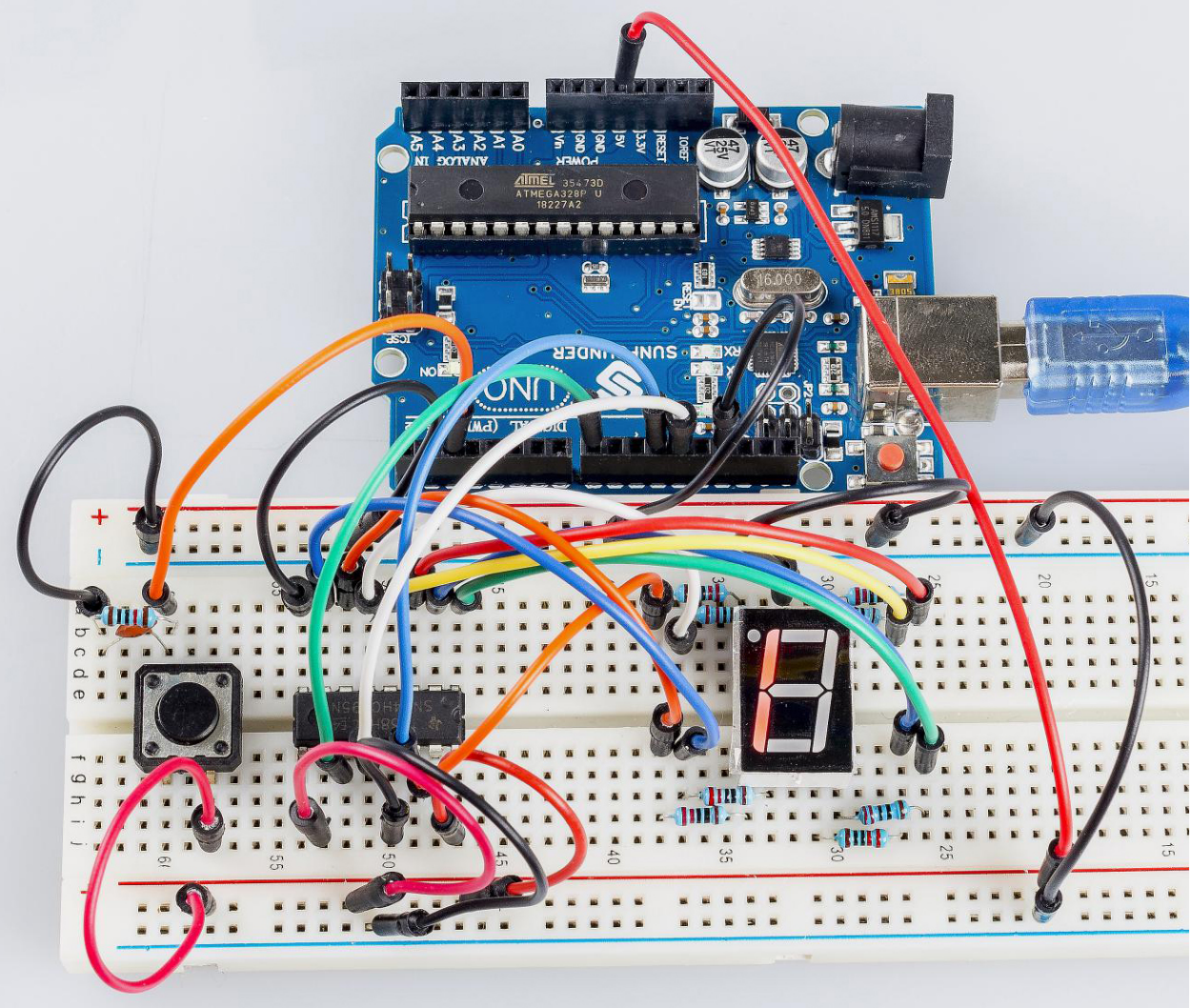


第 2 步：打开代码文件 `Lesson_23_Digital_Dice.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

你现在可以看到 7 段数码管循环显示 1~6。按下按键，显示速度会减慢，直到三秒后停止。再次按下按键，该过程将重复。



6.23.5 代码

6.23.6 代码分析

初始随机数来自 A0

```
randomSeed(analogRead(0));
```

初始随机数是从 A0 生成的，随机数的范围是 0-1023。

数字骰子

```
void loop()
{
    int stat = digitalRead(keyIn);  //store value read from keyIn
    if(stat == HIGH)  // check if the pushbutton is pressed
```

如果是，相应的引脚为高电平。

```
{
    num ++; // num adds 1
    if(num > 1)
    {
        num = 0;
    }
}
```

如果 num > 1，则清除该值。这是为了防止重复按压。所以不管你按多少次都算一次。

```
Serial.println(num);  // print the num on serial monitor
if(num == 1)  //when pushbutton is pressed
{
    randNumber = random(1,7); //Generate a random number in 1-7
    showNum(randNumber);  //show the randNumber on 7-segment
    delay(1000);  //wait for 1 second
    while(!digitalRead(keyIn));  //When not press button,program stop here.
```

让它一直显示最后一个随机数。

```
int stat = digitalRead(keyIn);
```

再次读取按键的状态。

```
if(stat == HIGH) // check if the pushbutton is pressed
```

如果是，请运行下面的代码。

```
{
        num ++; // num+1=2
        digitalWrite(ledPin,HIGH); //turn on the led
        delay(100);
        digitalWrite(ledPin,LOW); //turn off the led
        delay(100);
        if(num >= 1) // clear the num
        {
            num = 0;
        }
    }
    }
    //show random numbers at 100 microseconds intervals
    //If the button has not been pressed
    randNumber = random(1,7);
    showNum(randNumber);
    delay(100);
}
```

showNum() 函数

```
void showNum(int num)
{
    digitalWrite(latchPin,LOW); //ground latchPin and hold low for transmitting
    shiftOut(dataPin,clockPin,MSBFIRST,datArray[num]);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(latchPin,HIGH); //pull the latchPin to save the data
}
```

该功能是在 7 段数码管上显示 `dataArray[]` 中的数字。

CHAPTER 7

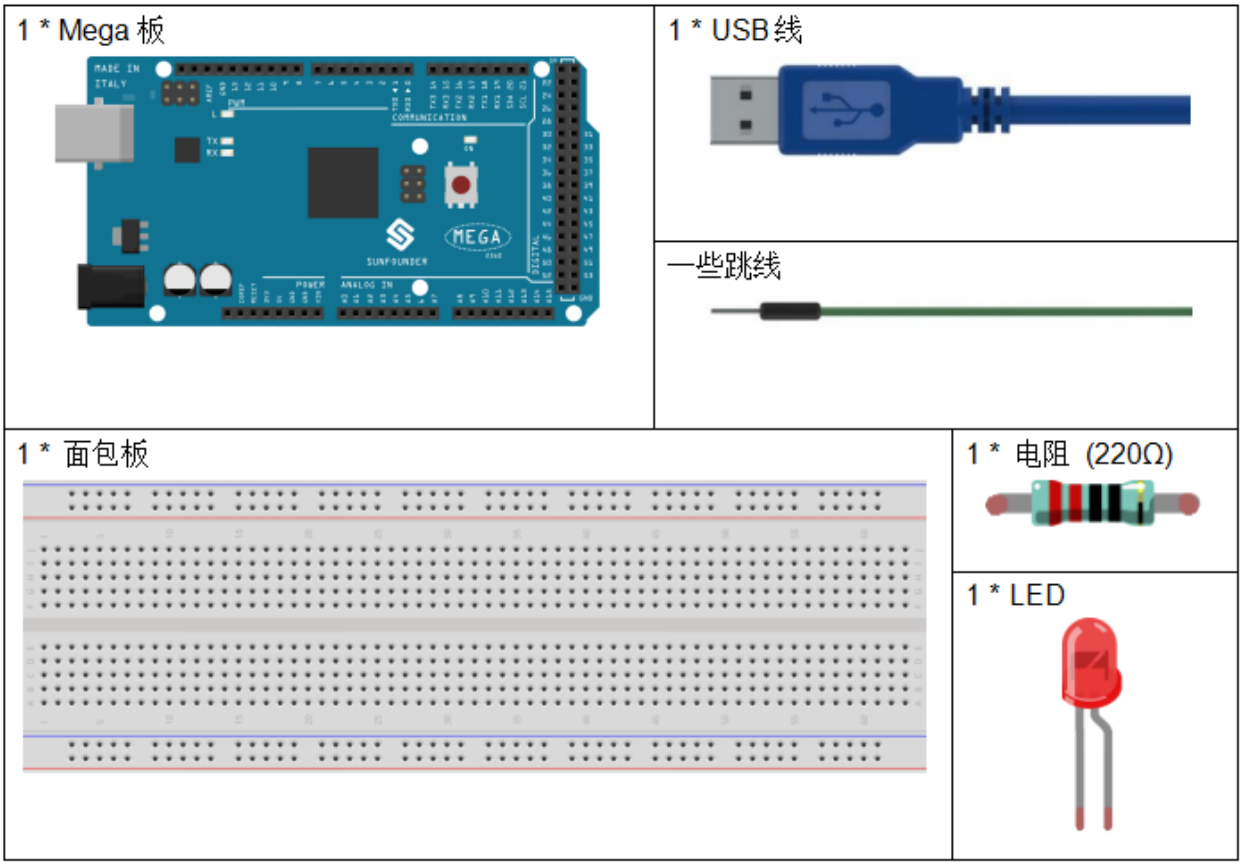
Arduino 项目 - Mega2560

7.1 第 1 课闪烁的 LED

7.1.1 介绍

你之前应该已经学会了如何安装 Arduino IDE 并添加库。现在你可以从一个简单的实验开始，学习 IDE 中的基本操作和代码。

7.1.2 所需器件

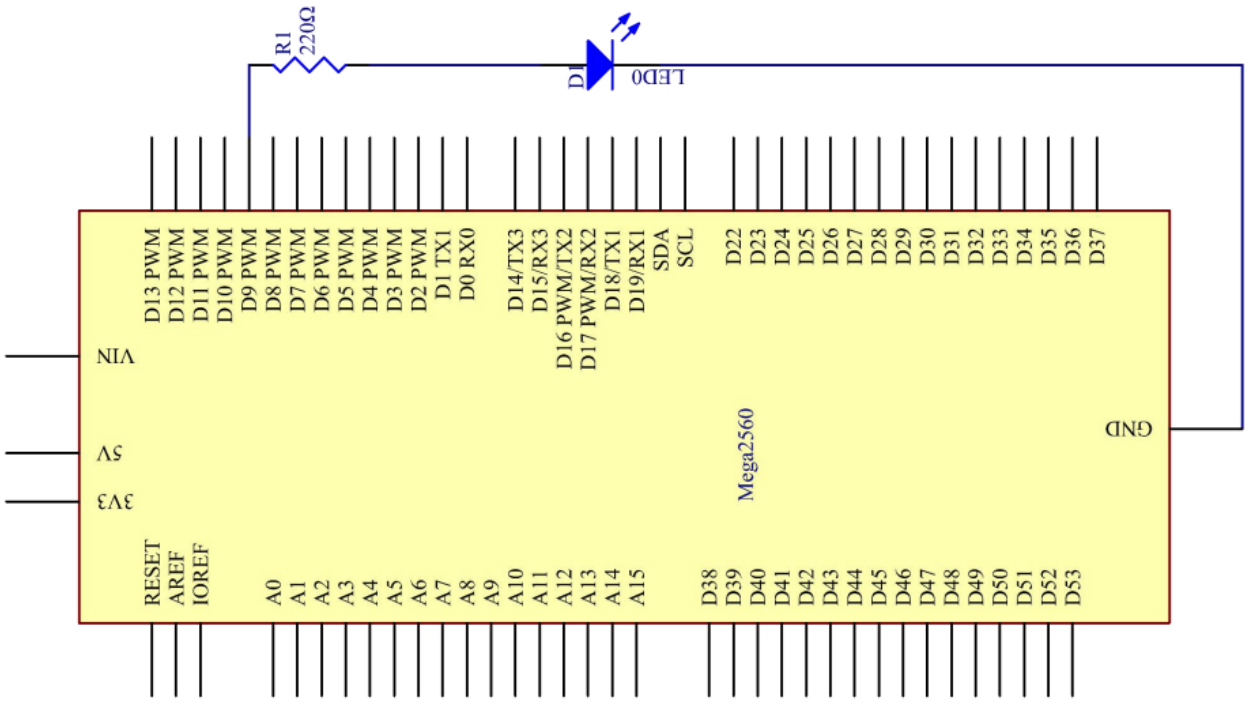


- *SunFounder Mega 板*
- *面包板*
- *跳线*
- *LED 发光二极管*
- *电阻*

7.1.3 原理图

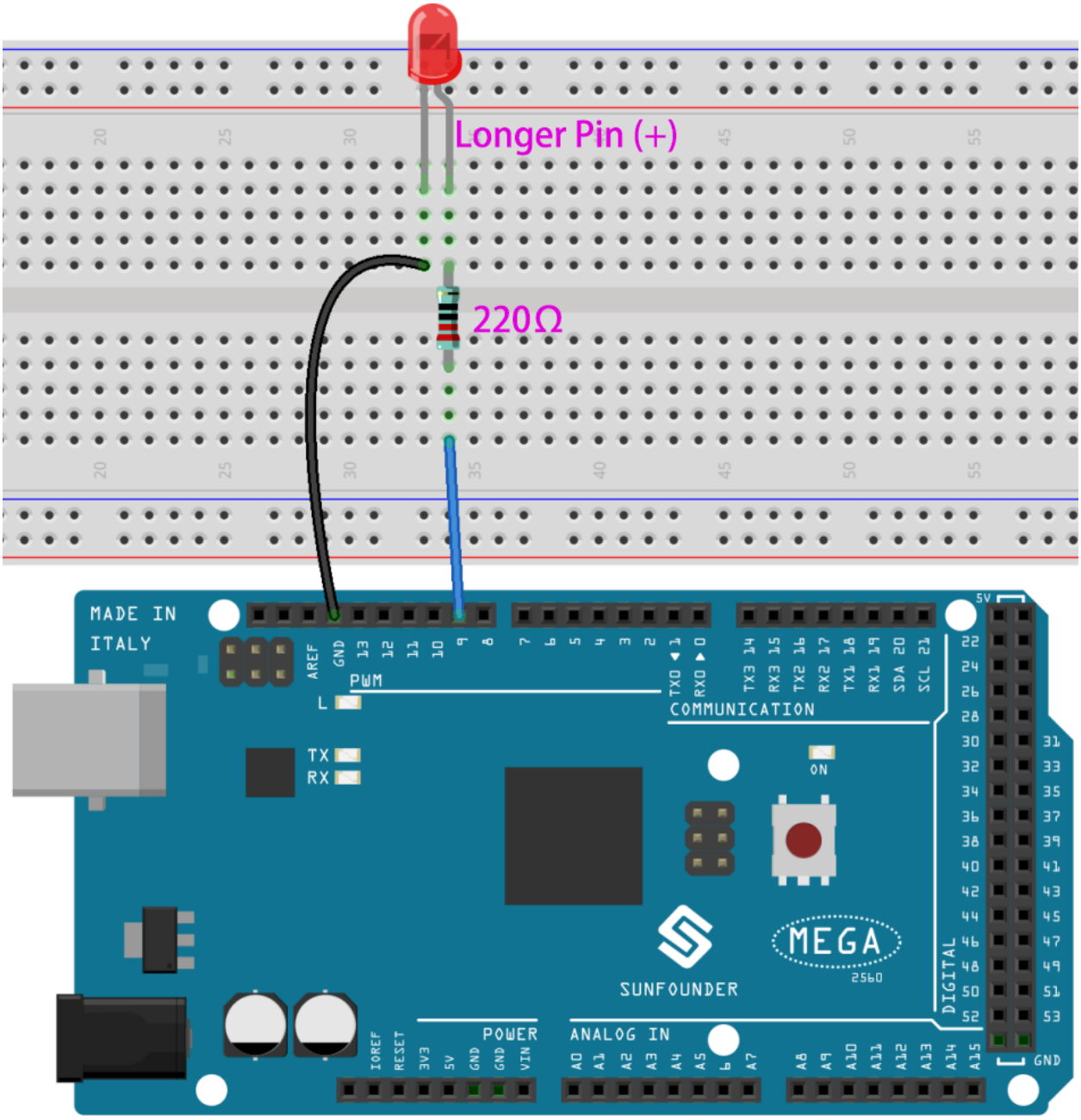
将 220ohm 电阻的一端接板子的 9 脚，另一端接 LED 的正极（长脚），LED 的负极（短脚）接 GND。当 9 脚输出高电平时，电流通过限流电阻到达 LED 的正极。由于 LED 的阴极连接到 GND，因此 LED 会亮起。当引脚 9 输出低电平时，LED 熄灭。

原理图如下所示：



7.1.4 实验步骤

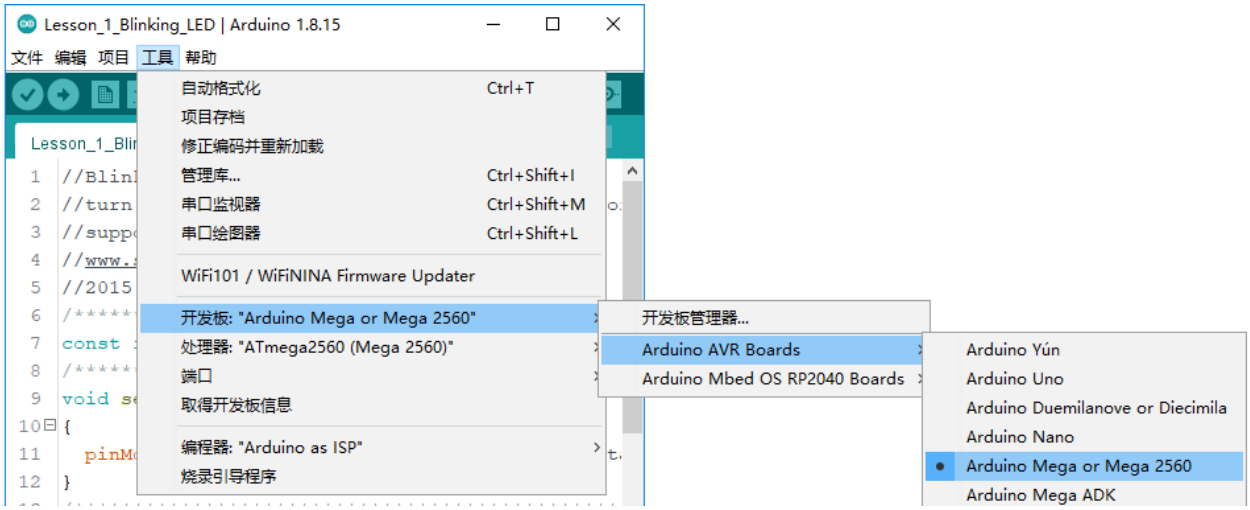
第 1 步：搭建电路。(LED 上长的引脚为阳极)。然后用一根 USB 线将板子插入到电脑上。



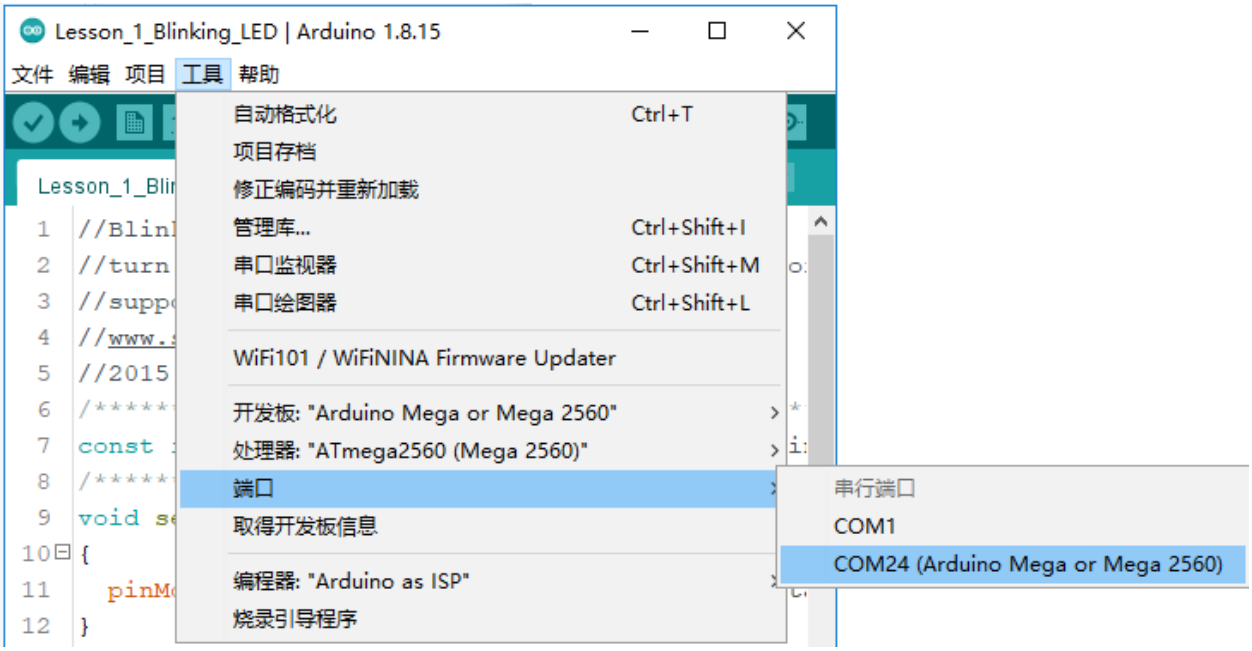
第 2 步：打开路径 `SunFounder Uno R3 学习套件\Arduino 项目代码\Lesson_1_Blinking_LED` 中的代码文件 `Lesson_1_Blinking_LED.ino`。

第 3 步：在上传代码前，你需要选择正确的开发板和端口。

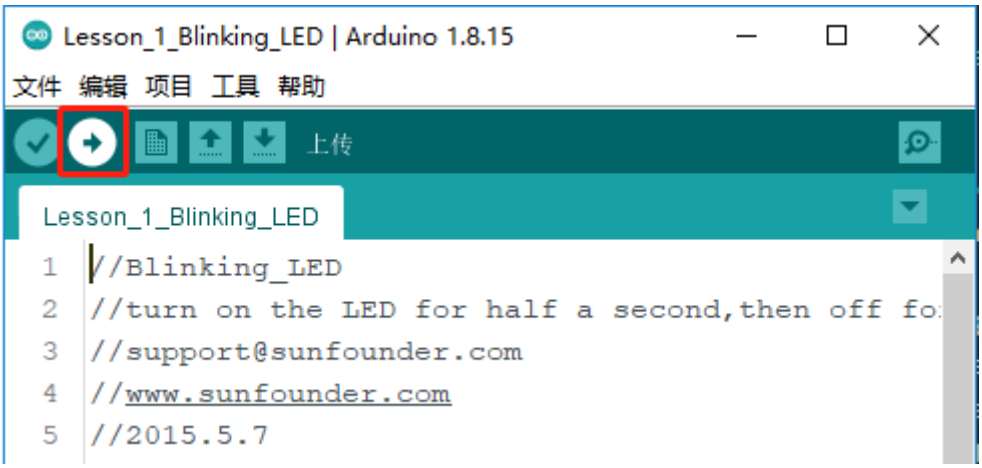
点击 **工具 -》开发板**，然后选择 **Arduino/Genuino Mega or Mega 2560**。



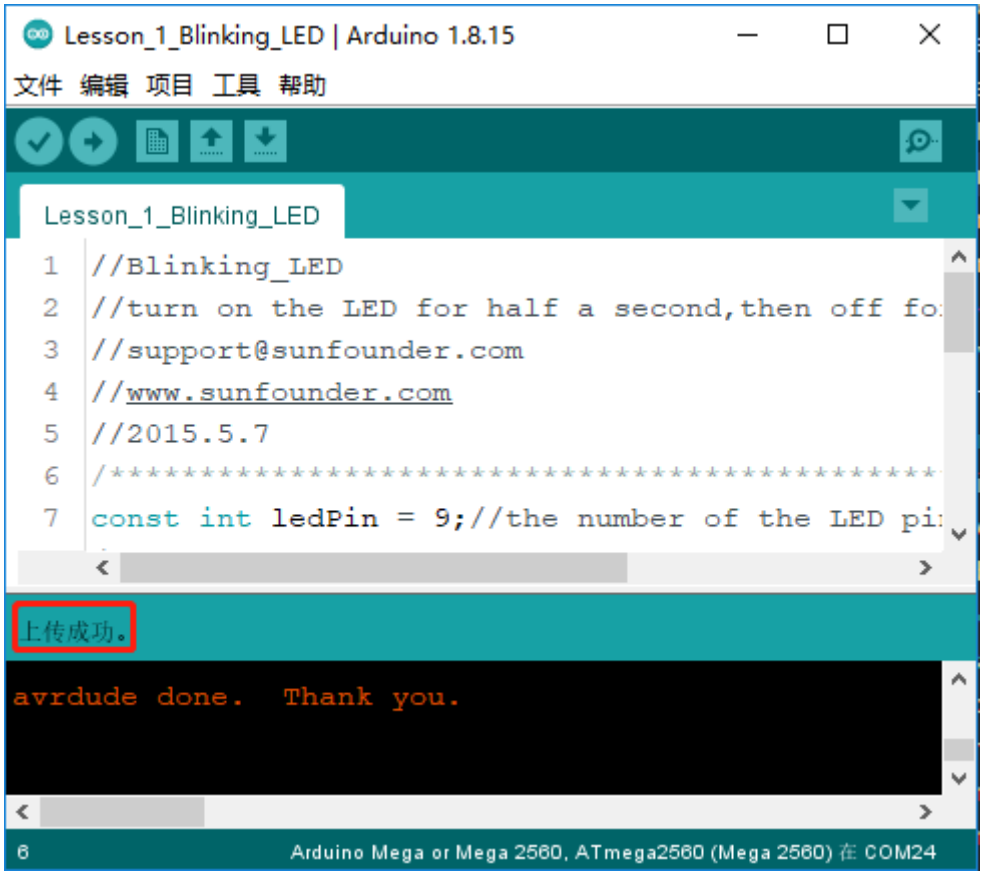
然后再点击 **工具 -》端口**，你的端口应该和我的不一样。



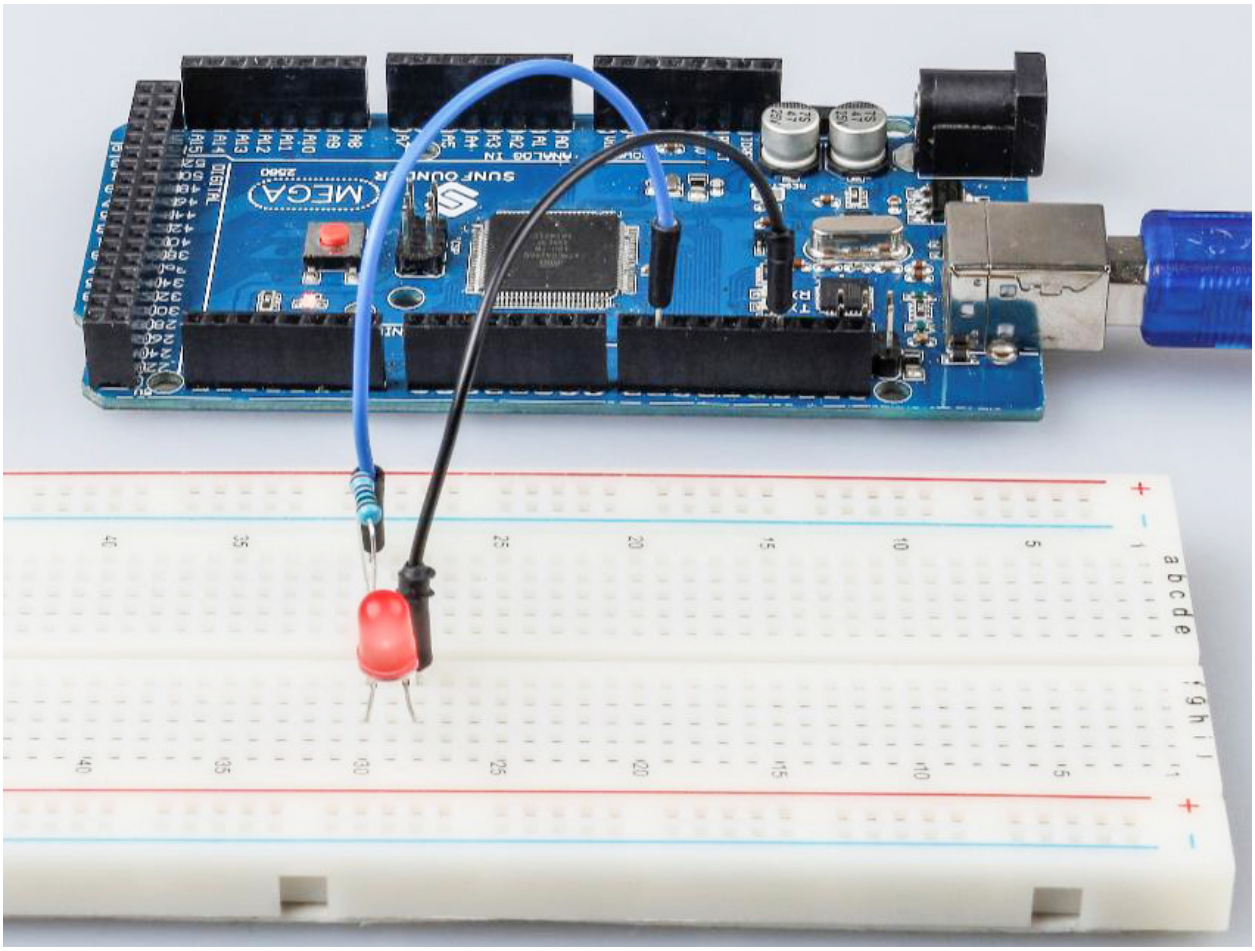
第 4 步：点击 **上传**按钮来将代码上传到主板上。



第 5 步：如果 **上传成功**提示出现，代表代码已成功上传到主板上。



现在你将看到 LED 闪烁。



7.1.5 代码

7.1.6 代码分析

定义变量

```
const int ledPin = 9; //the number of the LED pin
```

你应该在使用前定义每个变量，以防出错。该行为引脚 9 定义了一个常量变量 ledPin。在下面的代码中，ledPin 代表引脚 9。你也可以直接使用引脚 9 代替。

setup() 函数

一个典型的 Arduino 程序由两个子程序组成：用于初始化的 `setup()` 和包含程序主体的 `loop()`。

- `setup()`：该函数通常用于初始化数字引脚，并将它们设置为输入或输出，以及串行通信的波特率。
- `loop()`：该函数包含了整个代码运行顺序，将循环运行，除非发生停电之类的事情，否则它不会停止。

```
void setup()
{
    pinMode(ledPin,OUTPUT);//initialize the digital pin as an output
}
```

在 `setup()` 函数中将 `ledPin` 设置为输出。

- `pinMode(Pin)`：将指定的引脚配置为输入或输出。

setup 之前的 void 意味着这个函数不会返回值。即使不需要初始化引脚，你仍然需要此功能。否则编译会出错。

loop() 函数

```
void loop()
{
    digitalWrite(ledPin,HIGH);//turn the LED on
    delay(500);               //wait for half a second
    digitalWrite(ledPin,LOW); //turn the LED off
    delay(500);               //wait for half a second
}
```

本程序是设置 `ledPin` 为 `HIGH` 来让 LED 点亮，使用 `delay()` 函数来设置点亮时间，单位为毫秒。同样，设置为 `LOW` 将让 LED 熄灭，时间为 500 毫秒。代码上传之后，你将看到 LED 点亮 500 毫秒 (0.5 秒)，熄灭 500 毫秒（0.5 秒），这种交替不会停止，除非断电。

- `digitWrite()`：写一个 `HIGH` 或 `LOW` 值到数字引脚。当此引脚在 `pinModel()` 函数中设置为输出时，其电压将设置为相应的值：5V（或 3.3V 板上的 3.3V）代表高，0V（地）代表低。

7.1.7 实验总结

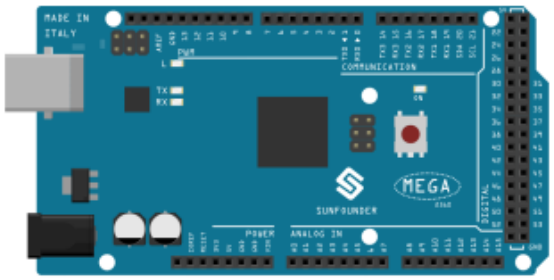
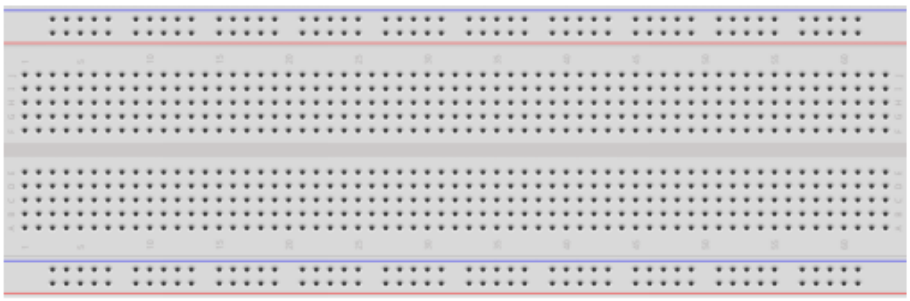
通过这个实验，你已经学会了如何打开 LED。你还可以通过更改 `delay (num)` 中的 num 值来更改 LED 的闪烁频率。例如，将其更改为 `delay(250)`，你会发现 LED 闪烁更快。

7.2 第 2 课流水灯

7.2.1 介绍

在本课中，我们将进行一个简单而有趣的实验——使用 LED 创造流动的 LED 灯。顾名思义，这 8 颗排成一排的 LED 依次亮起、变暗，就像流水一样。

7.2.2 所需器件

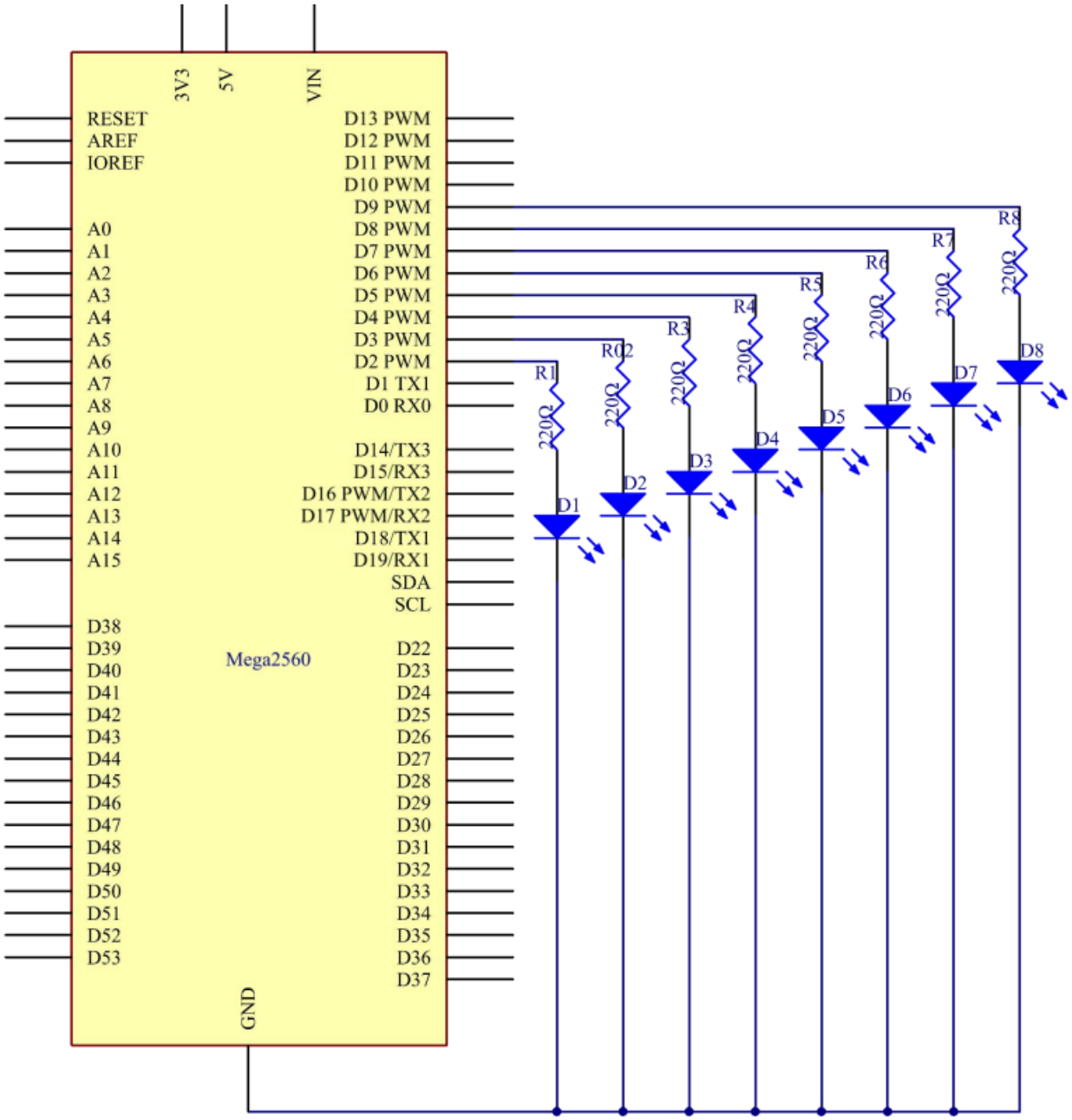
<table>
<tr><td>1 * Mega 板
  </td><td>1 * USB 线
 
一些跳线
  </td></tr>
<tr><td>1 * 面包板
  </td><td>8 * 电阻 (220Ω)
 
8 * LED
  </td></tr>
</table>

- *SunFounder Mega* 板
- 面包板
- 跳线
- *LED* 发光二极管
- 电阻

7.2.3 原理图

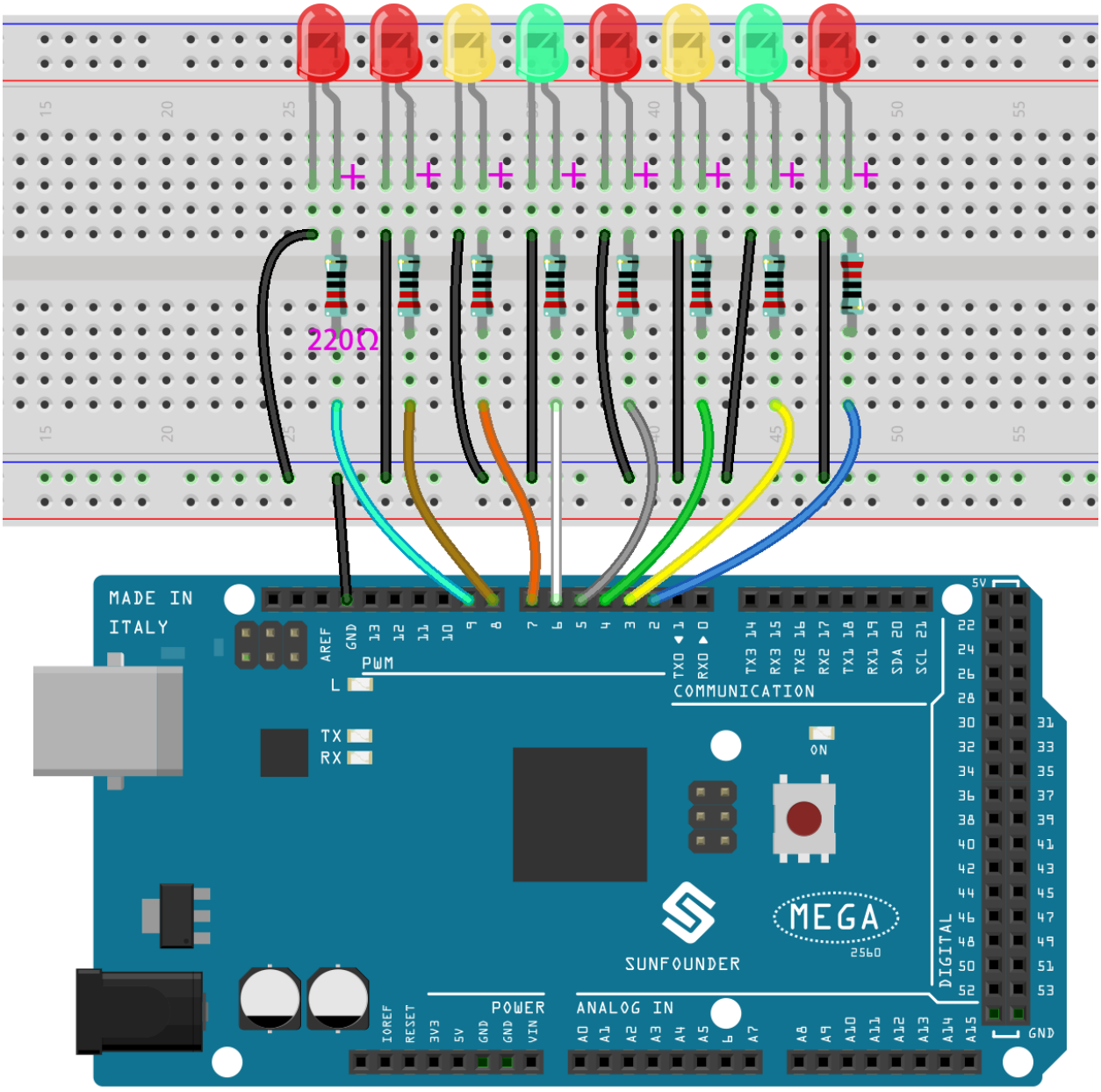
这个实验的原理很简单，就是依次点亮 8 个 LED。8 个 LED 分别连接到引脚 2~9。将它们设置为高电平，引脚上相应的 LED 将亮起。控制每个 LED 亮起的时间，你将看到流动的 LED 灯。

原理图如下所示：



7.2.4 实验步骤

第 1 步：搭建电路

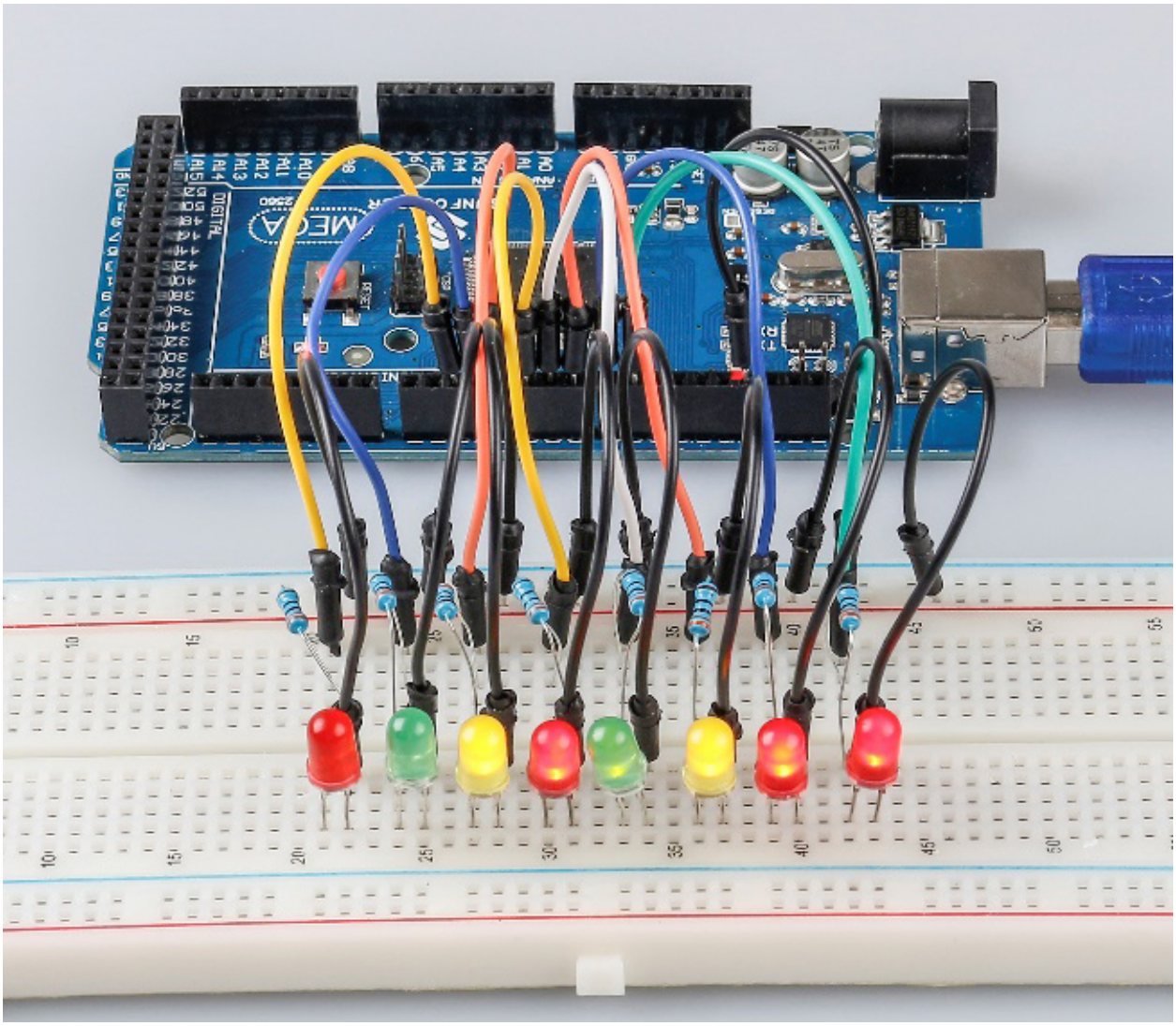


第 2 步：打开代码文件 `Lesson_2_Flowing_LED_Lights.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你应该看到八个 LED 灯从连接在第 2 脚的 LED 到第 9 脚的 LED 逐一变亮，然后从第 9 脚的 LED 到第 2 脚的 LED 依次变暗。这整个过程将重复进行，直到电路断电。



7.2.5 代码

7.2.6 代码分析

for() 语句

```
void setup()
{
    //set pins 2 through 9 as output
    for (int i = 2; i <= 9; i++)
    {
        pinMode(i, OUTPUT); //initialize a as an output
    }
}
```

`for (initialization; condition; increment) { //statement(s); }`: for 语句用于重复一个大括号内的语句块。初始化首先发生，而且正好一次。每次通过循环时，都要对条件进行测试；如果条件为真，语句块和增量就会被执行，然后再次测试条件。当条件变为假时，循环结束。

设置流水灯

使用 `for()` 语句将 2 引脚 ~9 引脚设置为高电平。

```
for (int a = 2; a <= 9; a++)
{
    digitalWrite(a, HIGH); //turn this led on
    delay(100); //wait for 100 ms
}
```

然后让 8 个 LED 依次从 9 引脚到 2 引脚熄灭。

```
for (int a = 9; a <= 2; a--)
{
    digitalWrite(a, LOW); //turn this led on
    delay(100); //wait for 100 ms
}
```

最后用同样的方法将 9 引脚到 2 引脚的 8 个 LED 依次点亮，让它们依次熄灭。

```
for (int a = 9; a <= 2; a--)
{
    digitalWrite(a, HIGH); //turn this led on
    delay(100); //wait for 100 ms
}
for (int a = 2; a <= 9; a++)
{
    digitalWrite(a, LOW); //turn this led on
    delay(100); //wait for 100 ms
}
```

7.2.7 实验总结

通过这个实验，你已经学会了如何使用 `for()` 语句，当你想缩短代码时，这是一个非常有用的语句。

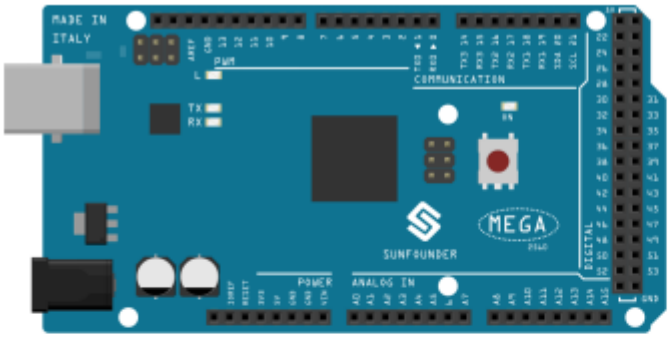
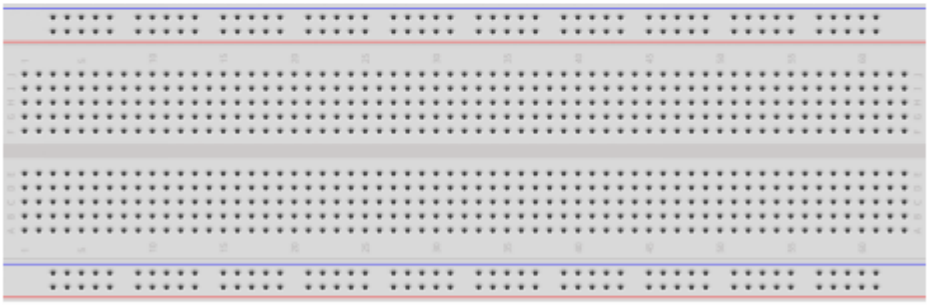
7.3 第 3 课按键

7.3.1 介绍

在本实验中，我们将学习如何使用 I/O 端口和按键来打开/关闭 LED。

“I/O 端口”是指 INPUT 和 OUTPUT 端口。这里使用控制板的 INPUT 端口来读取外部设备的输出。由于板子本身有一个 LED（连接到引脚 13），为了方便起见，你可以使用这个 LED 来做这个实验。

7.3.2 所需器件

<table>
<tr><td>1 * Mega 板
  </td><td>1 * 电阻 (10kΩ)
 
1 * 104 电容
  </td><td>1 * 按键
  </td></tr>
<tr><td colspan="3">1 * 面包板
  </td></tr>
<tr><td>1 * USB 线
  </td><td colspan="2">一些跳线
  </td></tr>
</table>

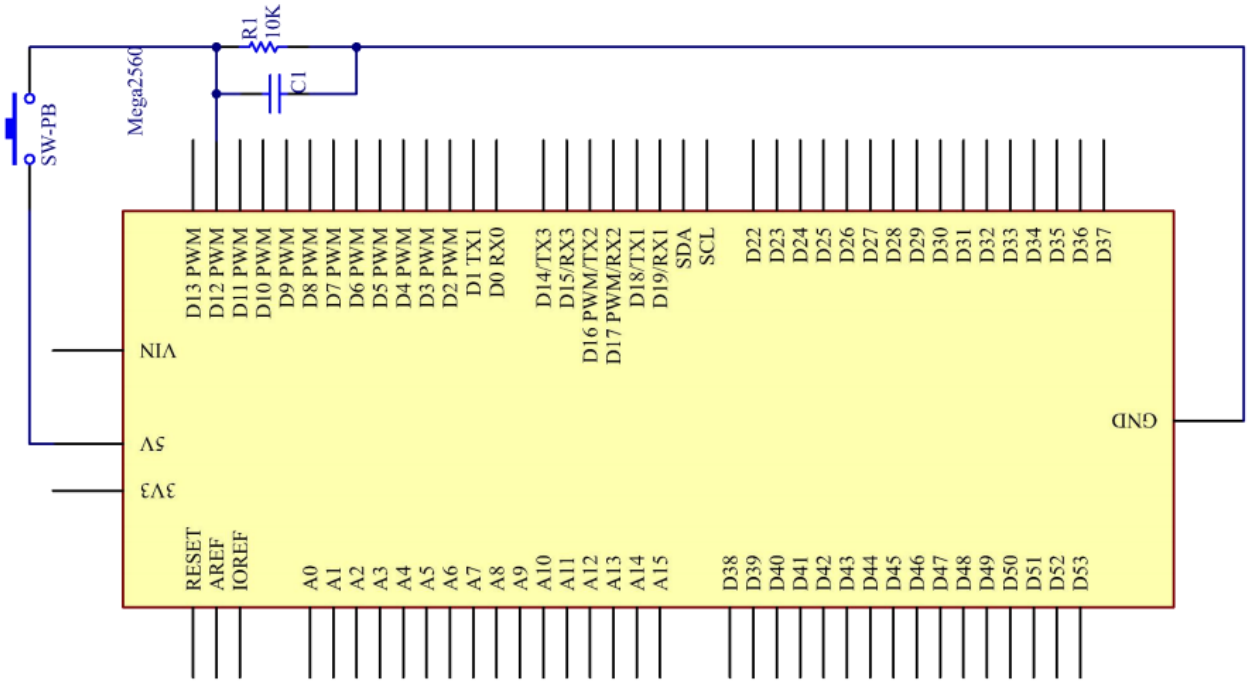
- *SunFounder Mega 板*
- 面包板
- 跳线
- 电阻
- 电容
- 按键

7.3.3 原理图

按键一端接 12 脚，同时接下拉电阻和 0.1uF（104）电容（用来消除抖动让按键工作时能够输出稳定电平）。

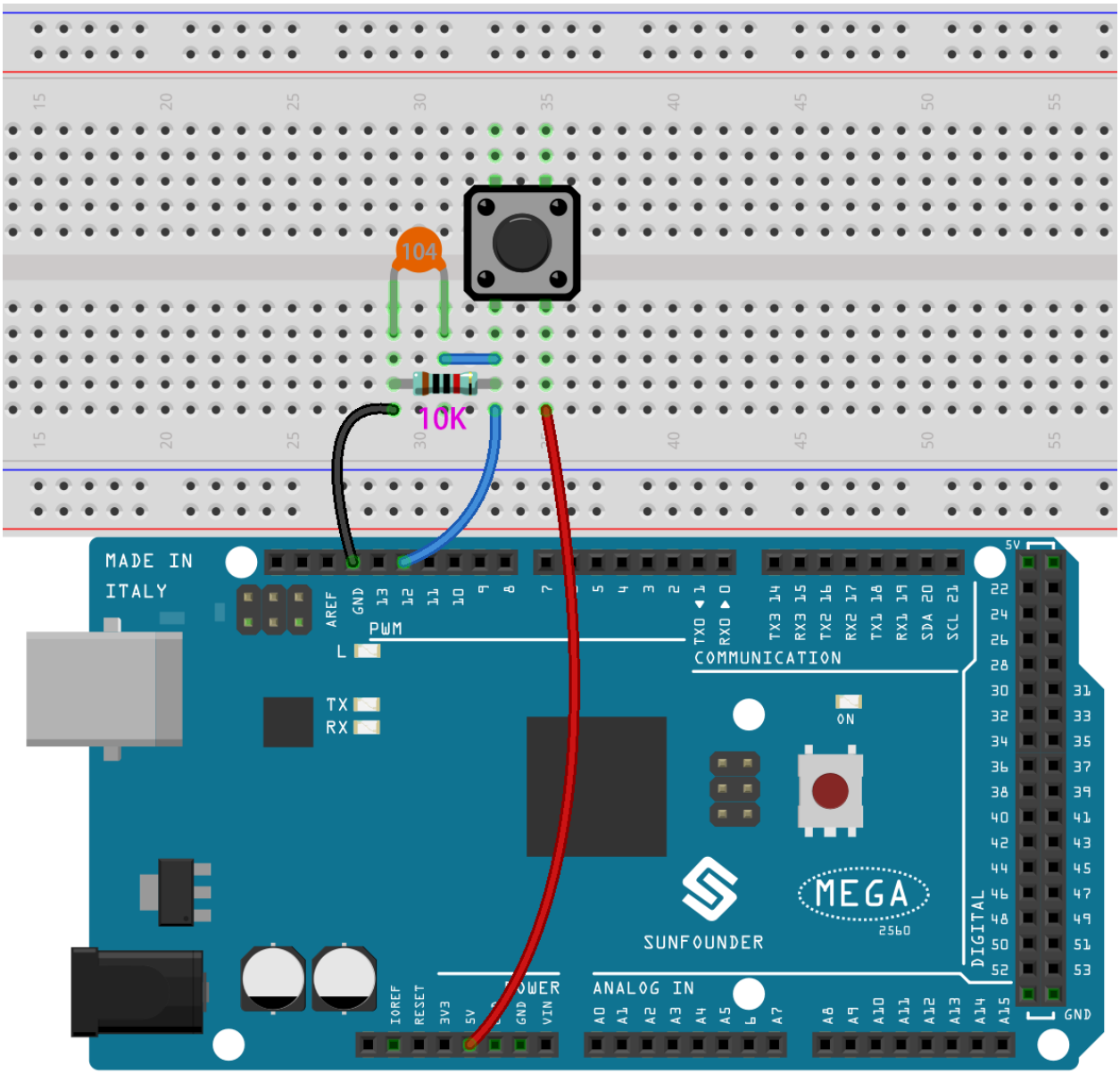
将电阻和电容的另一端连接到 GND，将按键另一端的一个引脚连接到 5V。按下按键时，引脚 12 为 5V（高电平），此时将 13 引脚设置为高电平来将控制板上的内置 LED 点亮。然后松开按键（引脚 12 变为低电平），引脚 13 为低电平。因此，我们将看到 LED 在按下和释放按键时交替亮起和熄灭。

原理图如下所示：



7.3.4 实验步骤

第 1 步：搭建电路。

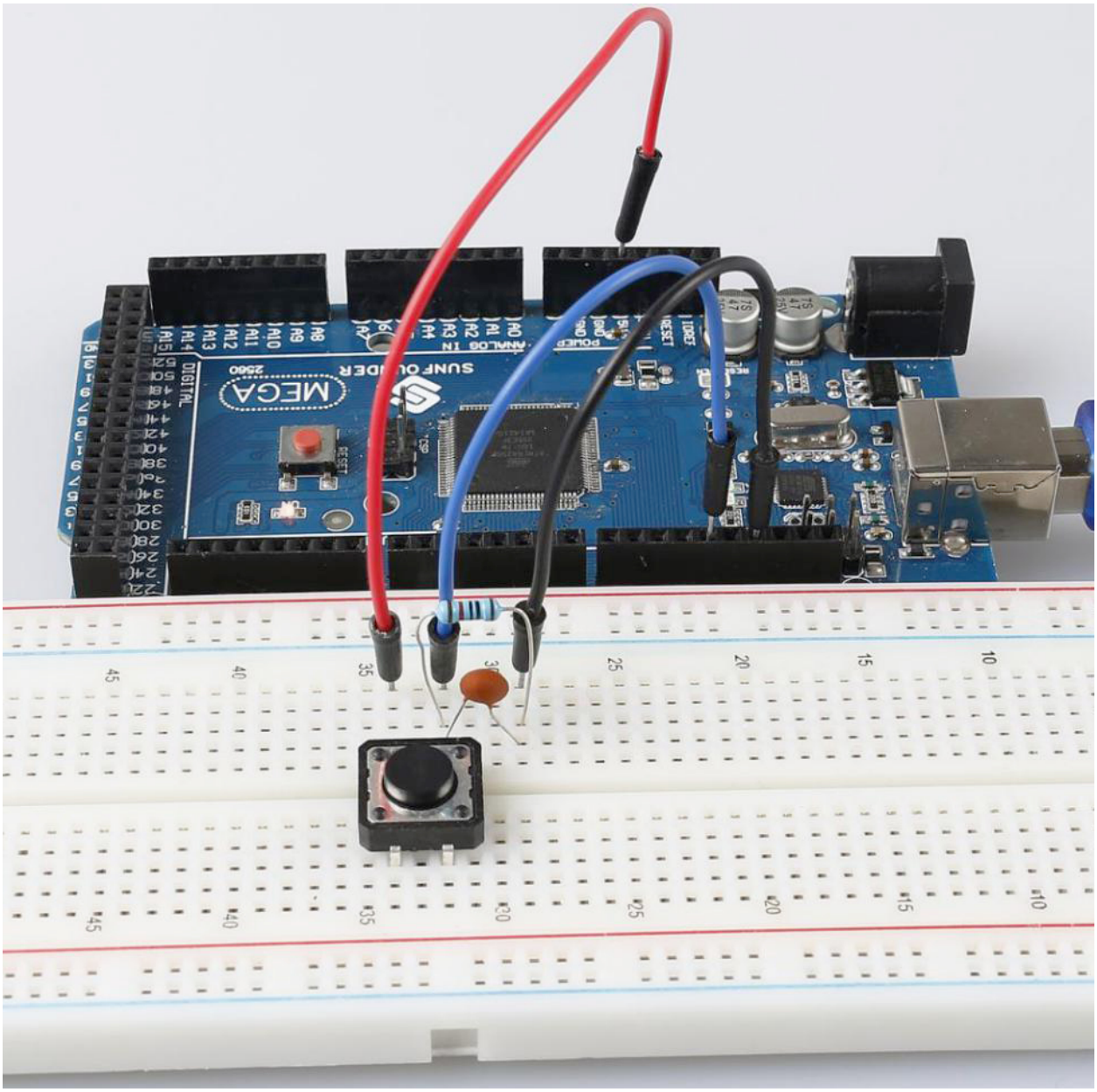


第 2 步：打开代码文件 `Lesson_3_Button.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在按下按键，控制板上的 LED 被点亮。



7.3.5 代码

7.3.6 代码分析

定义变量

```
const int buttonPin = 12; //the button connect to pin 12
const int ledPin = 13; //the led connect to pin13
int buttonState = 0; // variable for reading the pushbutton status
```

将按键连接到引脚 12，LED 已经连接到引脚 13。定义一个变量 `buttonState` 来存储按键的值。

设置引脚的输入输出状态

```
pinMode(buttonPin, INPUT); //initialize thebuttonPin as input
pinMode(ledPin, OUTPUT); //initialize the led pin as output
```

本次实验我们需要知道按键的状态，所以这里设置 buttonPin 为 INPUT ； 要设置 LED 的高/低，我们将 LedPin 设置为 OUTPUT。

读取按键状态

```
buttonState = digitalRead(buttonPin);
```

buttonPin``(Pin12) 是数字引脚；这里是读取按键的值并将其存储在 ``buttonState 中。

- digitalRead (Pin)：从指定的数字引脚读取值，无论是高电平还是低电平。

按键按下时让 LED 点亮

```
if (buttonState == HIGH )
{
    digitalWrite(ledPin, HIGH); //turn the led on
}
else
{
    digitalWrite(ledPin, LOW); //turn the led off
}
```

在这部分代码中，当 buttonState 为 HIGH 时，让 ledPin 为 HIGH ，LED 会被点亮。

由于按键的一端已连接至 5V，另一端已连接至引脚 12，因此按下按键时，引脚 12 为 5V（高电平）。然后用 if () 判断；如果条件为真，则 LED 将亮起。

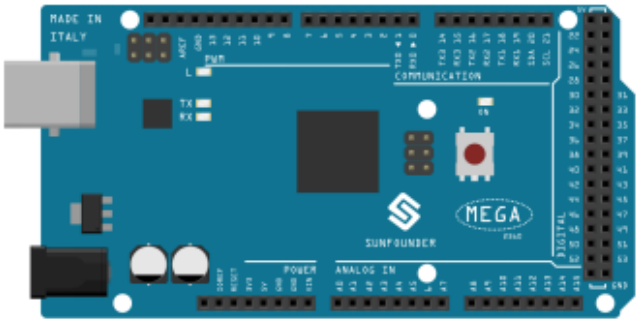
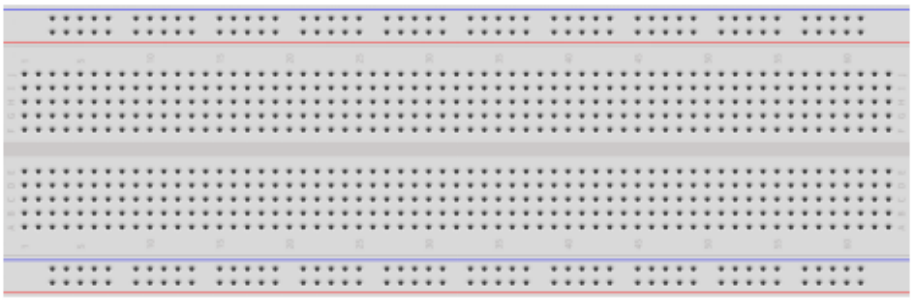
else 意味着当 if(conditional) 被确定为 false 时，运行 else。

7.4 第 4 课蜂鸣器

7.4.1 介绍

每当你想发出声音时，蜂鸣器都是你实验中的绝佳工具。在本课中，我们将学习如何驱动有源蜂鸣器来制作一个简单的门铃。

7.4.2 所需器件

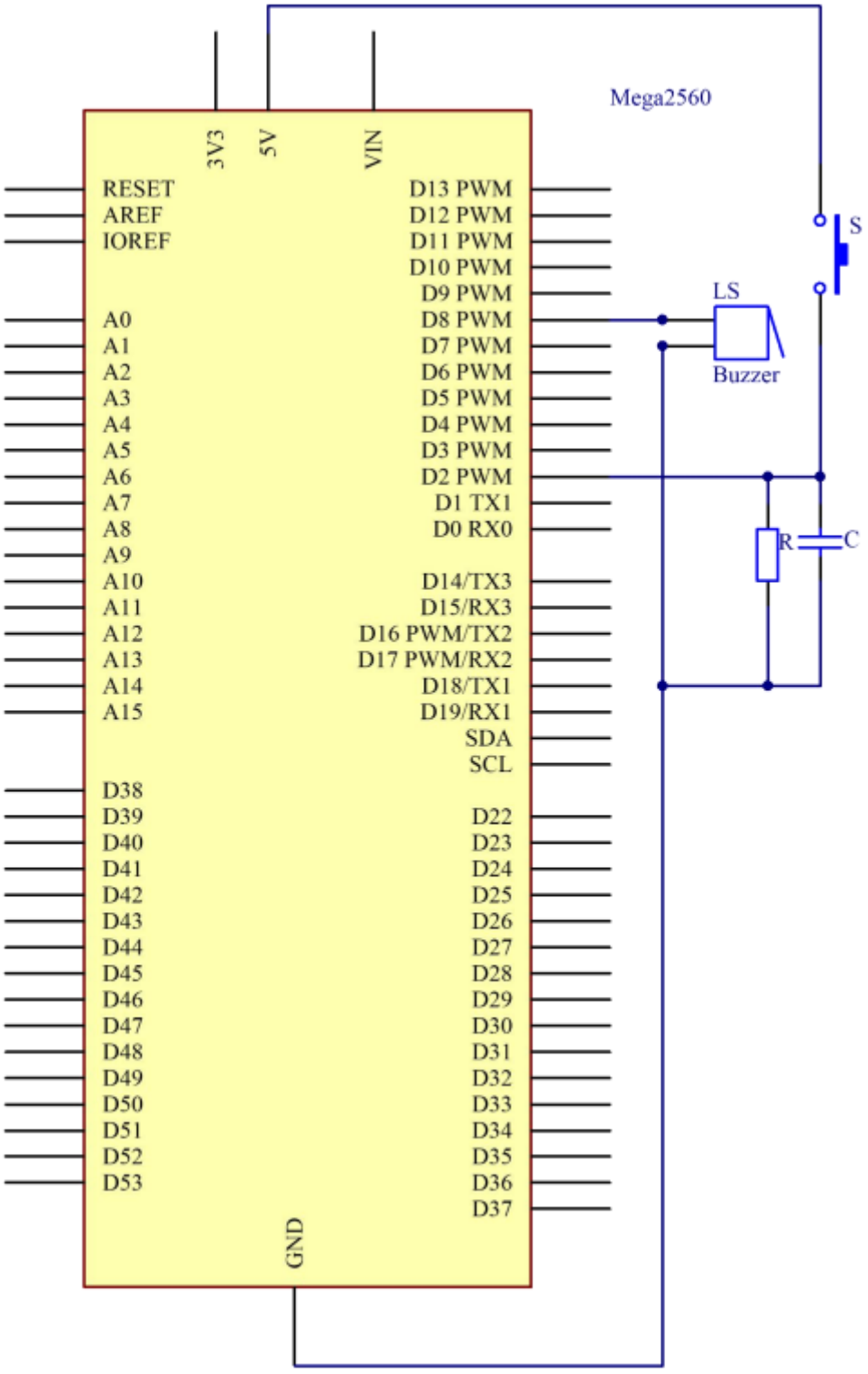
<table>
<tr><td colspan="2">1 * Mega 板
  </td><td>1 * 蜂鸣器 (有源)
  </td><td>1 * 104 电容
  </td></tr>
<tr><td colspan="2"></td><td>1 * 按键
  </td><td>1 * 电阻 (10kΩ)
  </td></tr>
<tr><td colspan="3">1 * 面包板
  </td></tr>
<tr><td colspan="2">1 * USB 线
  </td><td colspan="2">一些跳线
  </td></tr>
</table>

- *SunFounder Mega 板*
- 面包板
- 跳线
- 电阻
- 电容
- 按键
- 蜂鸣器

7.4.3 原理图

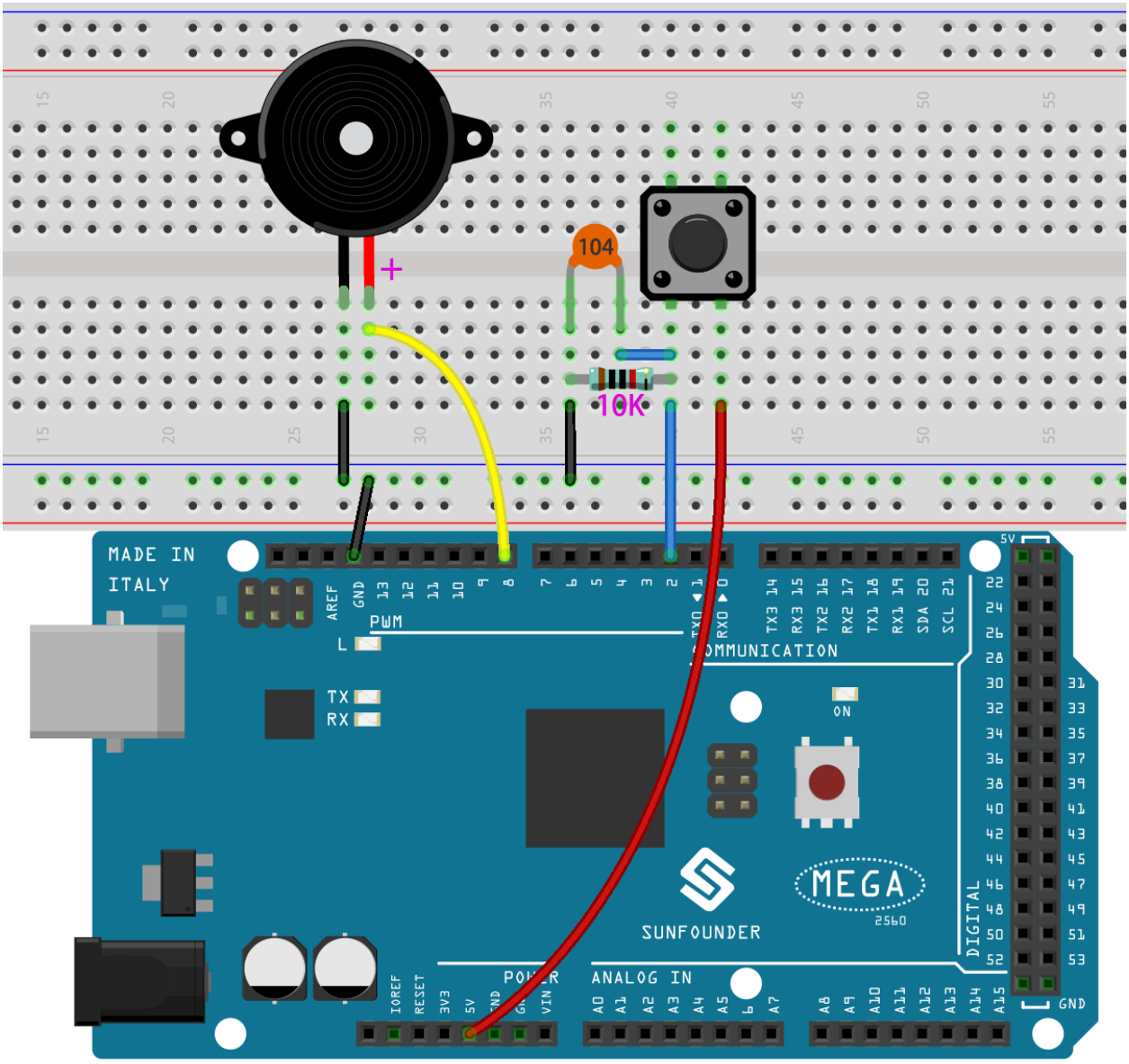
在这个课程中，使用的是有源蜂鸣器。

原理图如下所示：



7.4.4 实验步骤

第 1 步：搭建电路。（蜂鸣器长的引脚为阳极，短的为阴极）

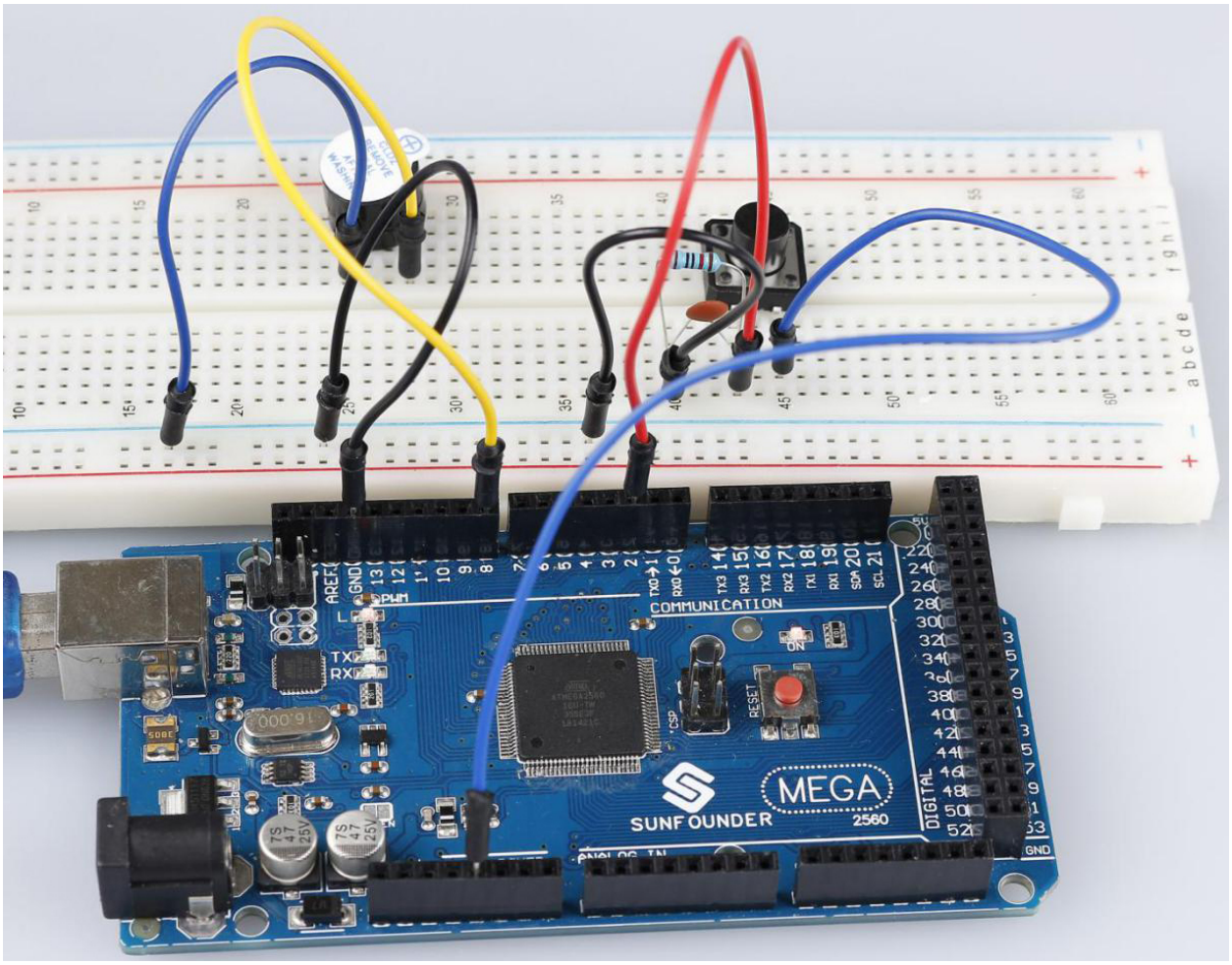


第 2 步：打开代码文件 `Lesson_4_Buzzer.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

按下按键，蜂鸣器将发出声音。



7.4.5 代码

7.4.6 代码分析

定义变量

```
const int buttonPin = 2; //the button connect to pin2
const int buzzerPin = 8; //the led connect to pin8
/*********************************/
int buttonState = 0; //variable for reading the pushbutton status
```

将按键连接到引脚 2，将蜂鸣器连接到引脚 8。定义一个变量 buttonState 来存储按键的值。

设置引脚的输入输出状态

```
void setup()
{
    pinMode(buttonPin, INPUT); //initialize the buttonPin as input
    pinMode(buzzerPin, OUTPUT); //initialize the buzzerpin as output
}
```

本次实验我们需要知道按键的状态，所以这里设置 buttonPin 为 INPUT；要设置蜂鸣器的高/低，我们将 buzzerPin 设置为 OUTPUT。

读取按钮状态

```
buttonState = digitalRead(buttonPin);
```

buttonPin (Pin2) 是数字引脚；这里是读取按钮的值并将其存储在 buttonState 中。

- digitalRead (Pin)：从指定的数字引脚读取值，无论是高电平还是低电平。

按下按钮让蜂鸣器发出声音

```
if (buttonState == HIGH ) //When press the button, run the following code.
{
    for (i = 0; i < 50; i++)
    /*When i=o, which accords with the condition i<=50, i++ equals to 1
    (here in i = i + 1, the two "i"s are not the same, but i(now) = i (before) + 1).
    Run the code in the curly braces: let the buzzer beep for 3ms and stop for 3ms.
    Then repeat 50 times.*/

    {
        digitalWrite(buzzerPin, HIGH); //Let the buzzer beep.
        delay(3); //wait for 3ms
        digitalWrite(buzzerPin, LOW); //Stop the buzzer.
        delay(3); //wait for 3ms
    }

    for (i = 0; i < 80; i++) //Let the buzzer beep for 5ms and stop for 5ms, repeat␣
↪80 times.
    {
        digitalWrite(buzzerPin, HIGH);
        delay(5); //wait for 5ms
        digitalWrite(buzzerPin, LOW);
        delay(5); //wait for 5ms
    }
}
```

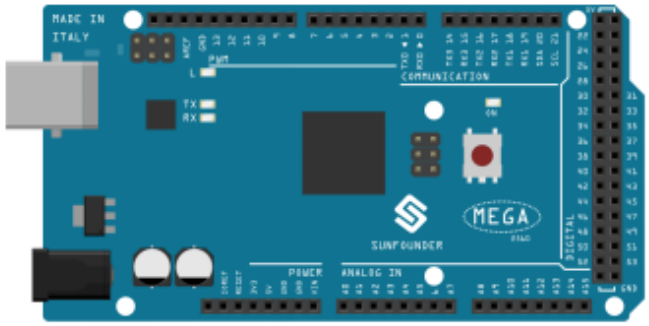
在这部分，当 buttonState 为高电平时，让蜂鸣器以不同的频率发出哔哔声，可以模拟门铃。

7.5 第 5 课倾斜开关

7.5.1 介绍

这里用的倾斜开关是内部有一个金属球，用来检测小角度的倾斜值。

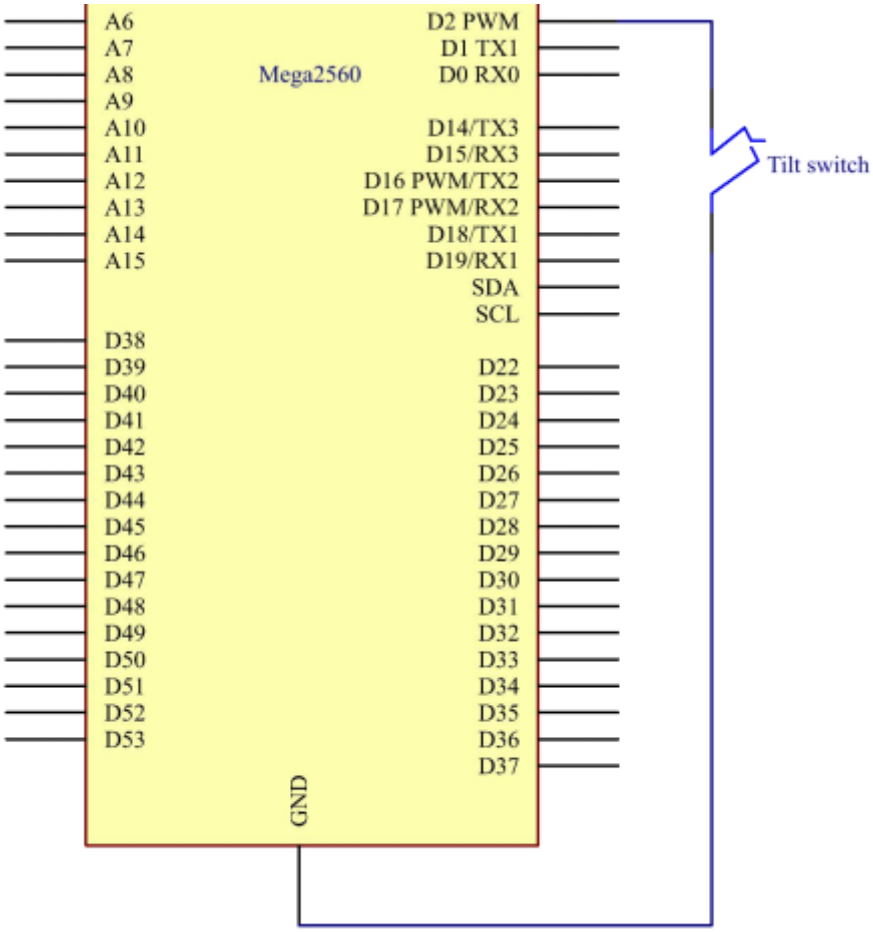
7.5.2 所需器件

<table>
<tr><td>1 * Mega 板
  </td><td>1 * 倾斜开关
  </td></tr>
<tr><td>1 * USB 线
  </td><td>2 * 杜邦线
  </td></tr>
</table>

- *SunFounder Mega 板*
- 面包板
- 跳线
- 倾斜开关

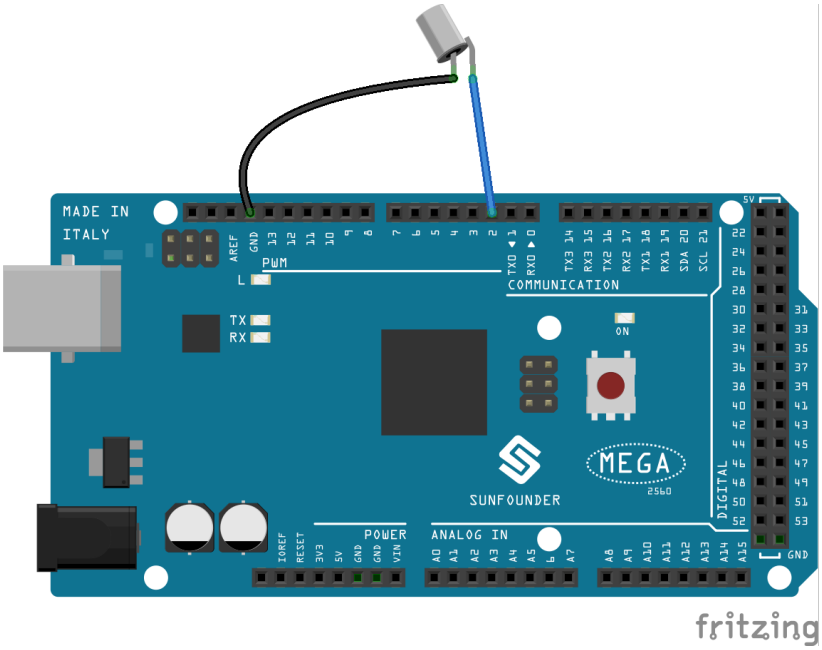
7.5.3 原理图

原理图如下图所示：



7.5.4 实验步骤

第 1 步：搭建电路。

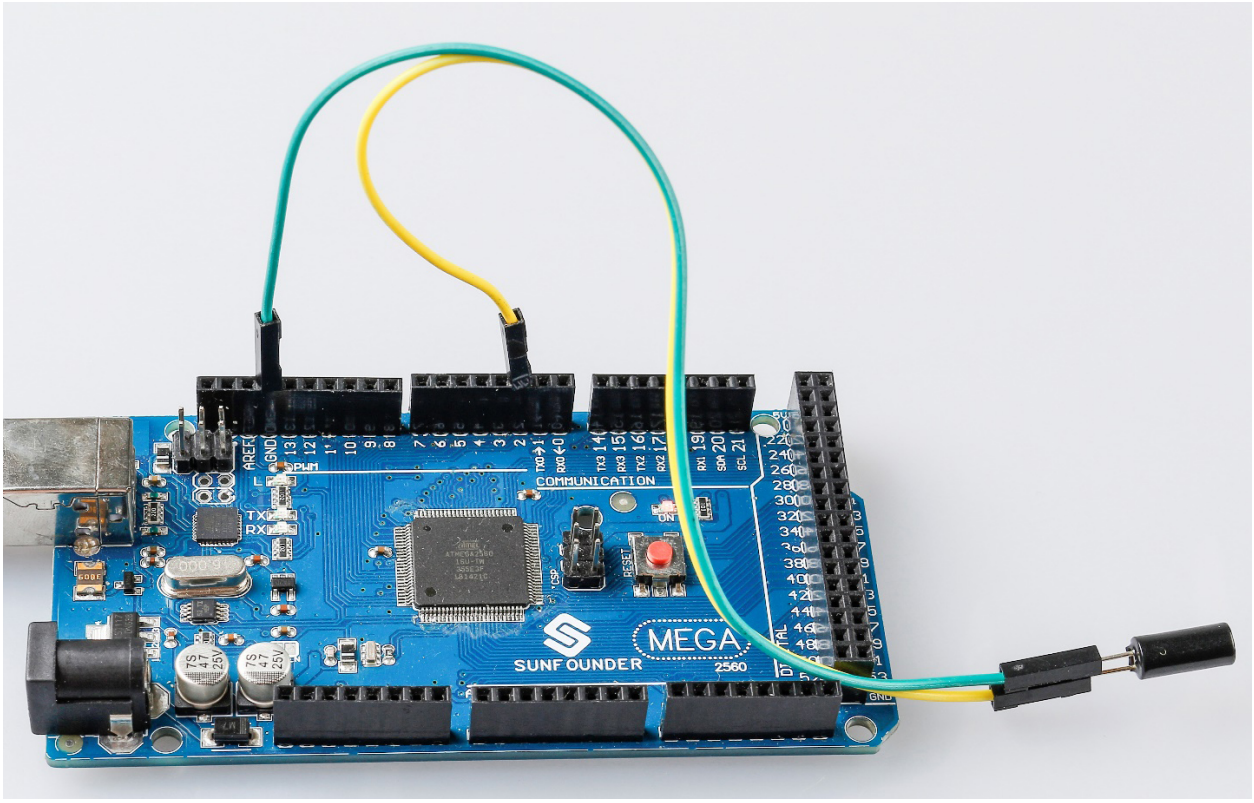


第 2 步：打开代码文件 `Lesson_5_Tilt_Switch.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，将开关倾斜，控制板上的 LED 将会被点亮。



7.5.5 代码

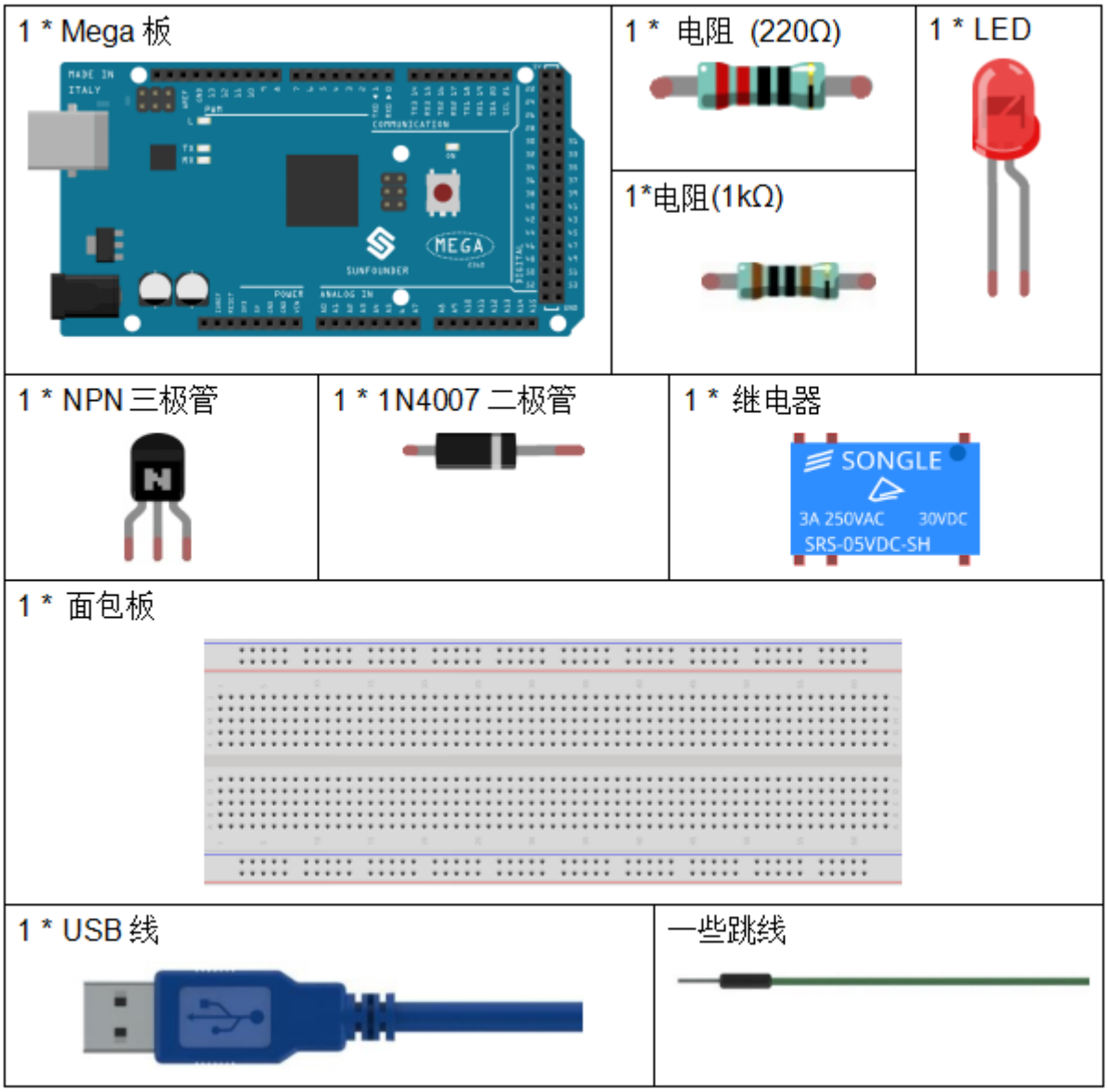
整个代码很简单，倾斜开关的一个脚接 pin2，另一脚接 GND，当倾斜开关时，开关的两个脚接 GND，然后让 pin13 上的 LED 点亮。

7.6 第 6 课继电器

7.6.1 介绍

我们可能知道，继电器是一种设备，用于响应所施加的输入信号在两个或多个点或设备之间提供连接。换句话说，继电器在控制器和设备之间提供隔离，因为设备可以在交流电和直流电下工作。然而，它们从工作于直流的微控制器接收信号，因此需要继电器来弥补差距。当你需要用小电信号控制大量电流或电压时，继电器非常有用。

7.6.2 所需器件

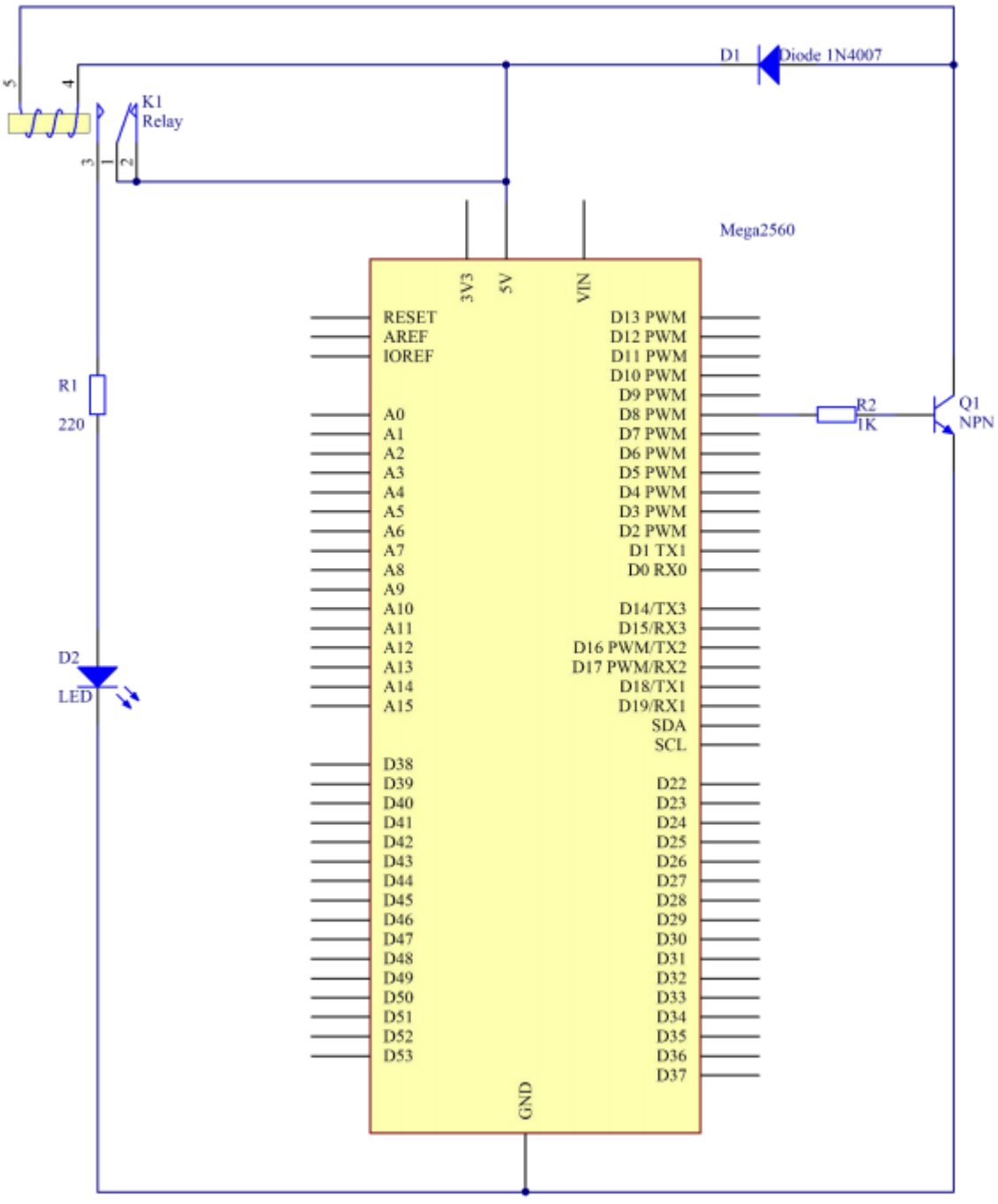


- *SunFounder Mega 板*
- 面包板
- 跳线
- 电阻
- 继电器
- 三极管
- 二极管
- *LED 发光二极管*

7.6.3 原理图

将 1K 电阻（晶体管通电时限流）连接到控制板的第 8 脚，然后连接到 NPN 晶体管，其集电极连接到继电器的线圈和发射极到 GND；将继电器的常开触点连接到 LED，然后连接到 GND。因此，当高电平信号提供给引脚 8 时，晶体管被激励，从而使继电器的线圈导通。然后它的常开触点闭合，LED 将亮起。当引脚 8 处于低电平时，LED 将熄灭。

原理图如下所示：



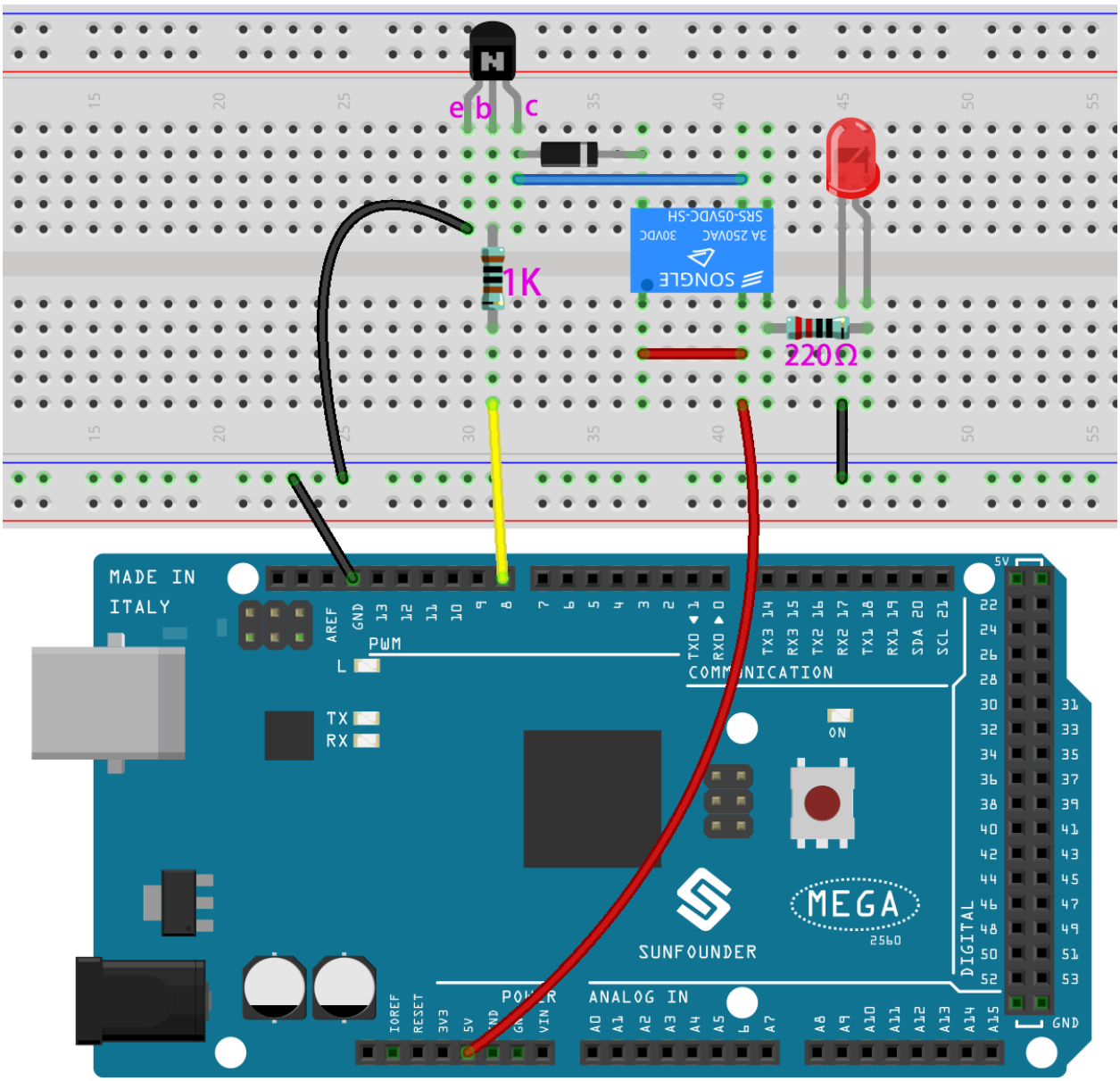
整流二极管的作用

当电压输入由 High（5V）变为 Low（0V）时，晶体管由饱和（放大、饱和、截止三种工作状态）变为截止，线圈中的电流突然无法流过。此时，如果没有整流二极管，线圈两端会产生反电动势（EMF），底部为正极，顶部为负极，电压高于 100V。这个电压加上来自晶体管电源的电压大到足以烧毁它。因此，整流二极管在将这个反电动势按上图箭头方向放电时极为重要，因此晶体管对 GND 的电压不高于 +5V（+0.7V）。

在本实验中，当继电器闭合时，LED 会亮起；当继电器打开时，LED 将熄灭。

7.6.4 实验步骤

第 1 步：搭建电路。

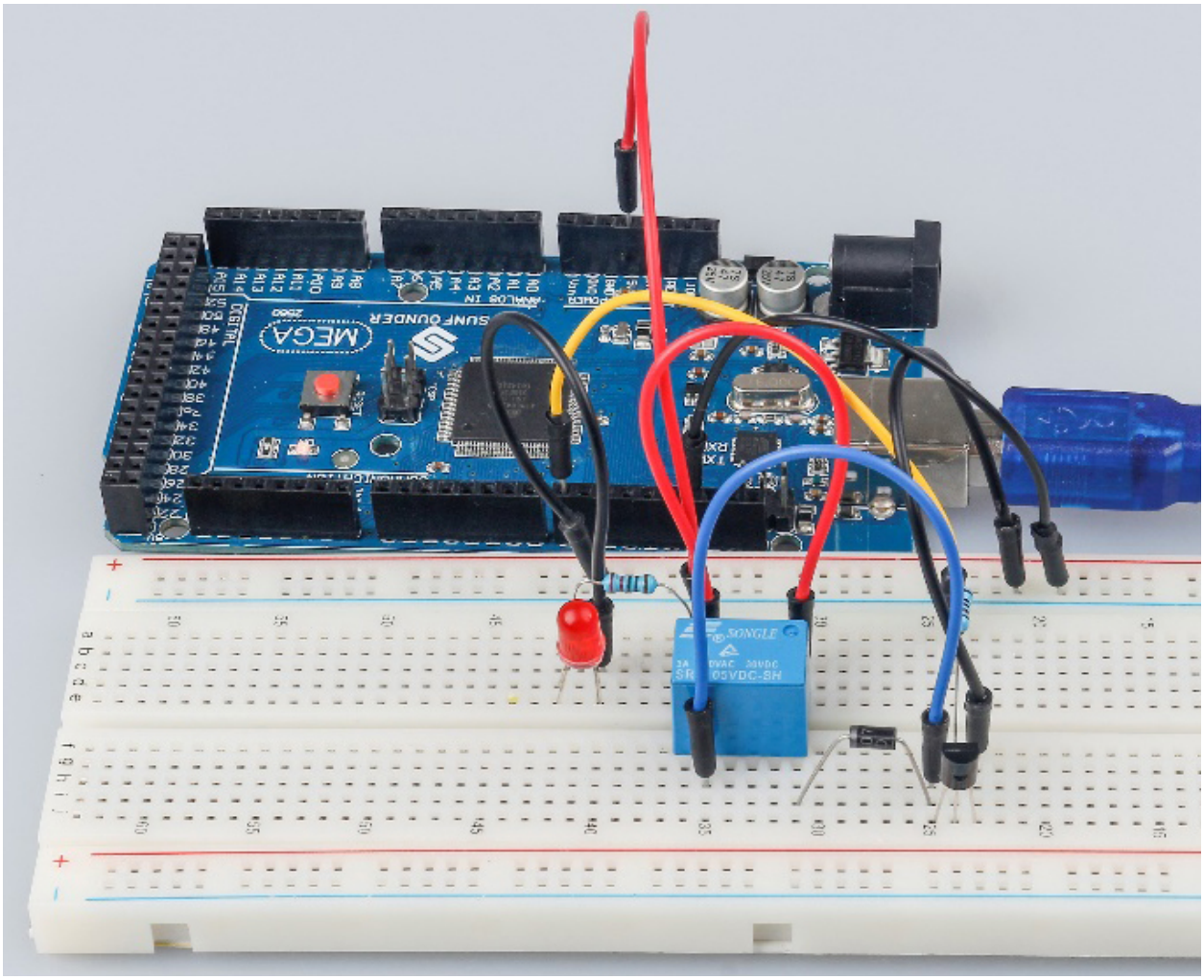


第 2 步：打开代码文件 `Lesson_6_Relay.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，发送一个高电平信号，继电器将关闭，LED 将亮起；发送一个低电平，它将打开并且 LED 将熄灭。此外，你还可以听到断开常闭触点并关闭常开触点引起的滴答声。



7.6.5 代码

7.6.6 代码分析

```
void loop()
{
    digitalWrite(relayPin, HIGH); //drive relay closure conduction
    delay(1000); //wait for a second
    digitalWrite(relayPin, LOW); //drive the relay is closed off
    delay(1000); //wait for a second
}
```

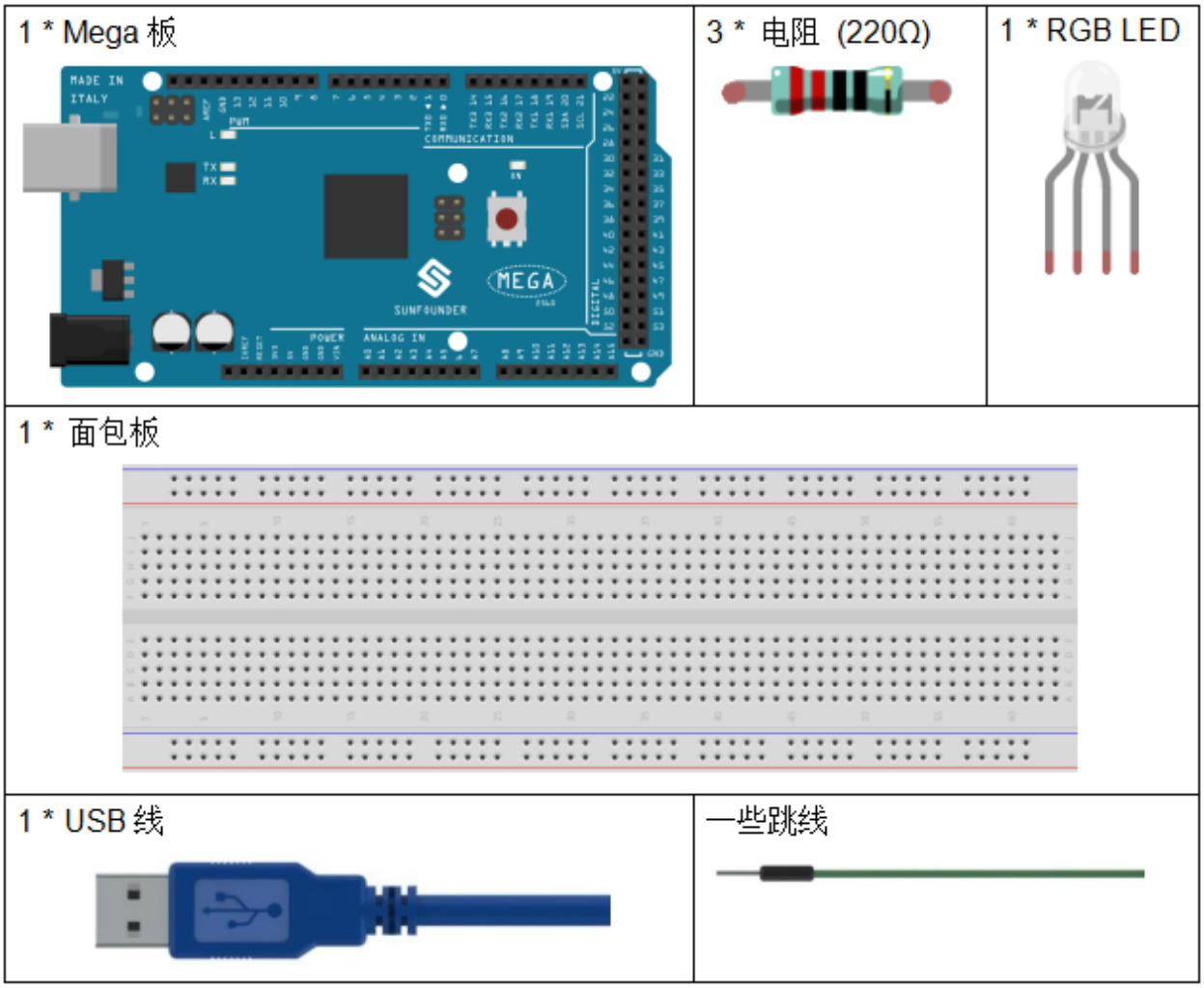
本实验中的代码很简单。首先，将 `relayPin` 设置为 `HIGH` 电平，连接到继电器的 LED 将亮起。然后将 `relayPin` 设置为低电平，LED 熄灭。

7.7 第 7 课 RGB LED

7.7.1 介绍

以前我们使用数字引脚来控制 LED 的亮度和亮度。在本课中，我们将使用 PWM 来控制 RGB LED 闪烁各种颜色。当 LED 的 R、G、B 引脚设置不同的 PWM 值时，其亮度会有所不同。当三种不同的颜色混合时，我们可以看到 RGB LED 闪烁不同的颜色。

7.7.2 所需器件

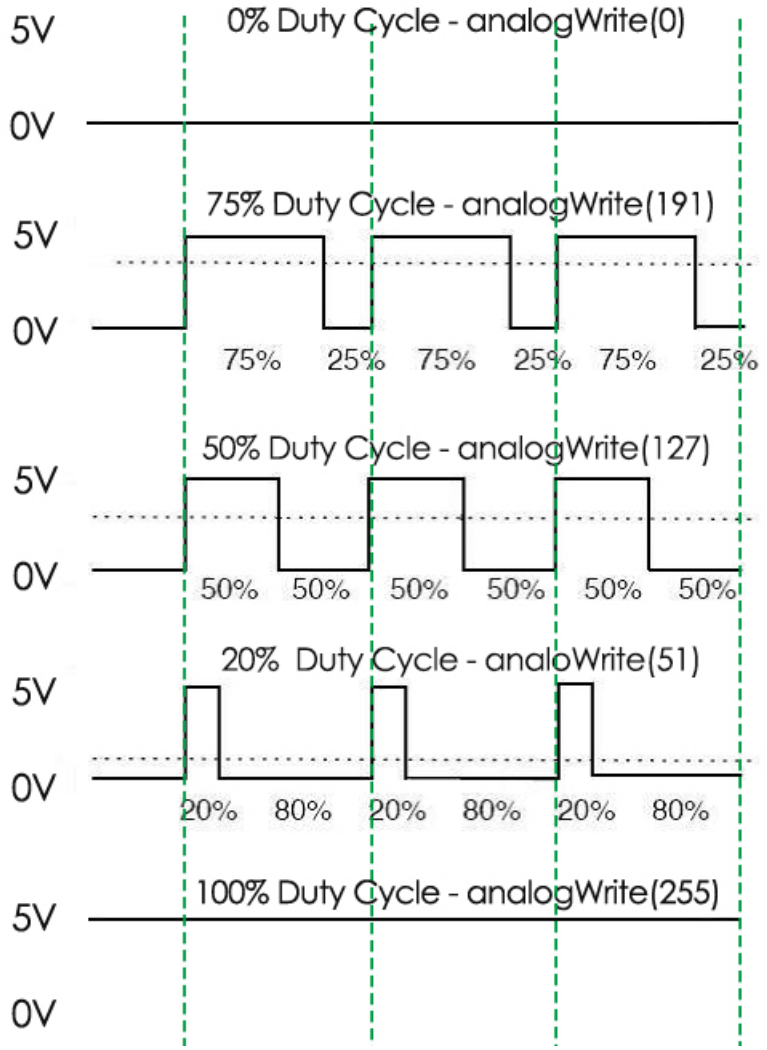


- *SunFounder Mega 板*
- 面包板
- 跳线
- 电阻
- *RGB LED*

7.7.3 PWM

脉宽调制或 PWM 是一种通过数字方式获得模拟结果的技术。数字控制用于创建方波，这是一种在开和关之间切换的信号。这种开关模式可以通过改变信号打开的时间部分与信号关闭的时间来模拟完全打开（5 伏）和关闭（0 伏）之间的电压。“导通时间”的持续时间称为脉冲宽度。要获得不同的模拟值，你可以更改或调制该宽度。如果你使用某些设备（例如 LED）足够快地重复这种开关模式，它会是这样的：信号是 0 到 5V 之间的稳定电压，用于控制 LED 的亮度。

在下图中，绿线代表一个固定的时间段。该持续时间或周期是 PWM 频率的倒数。换句话说，当 Arduino 的 PWM 频率约为 500Hz 时，每条绿线的测量值为 2 毫秒。



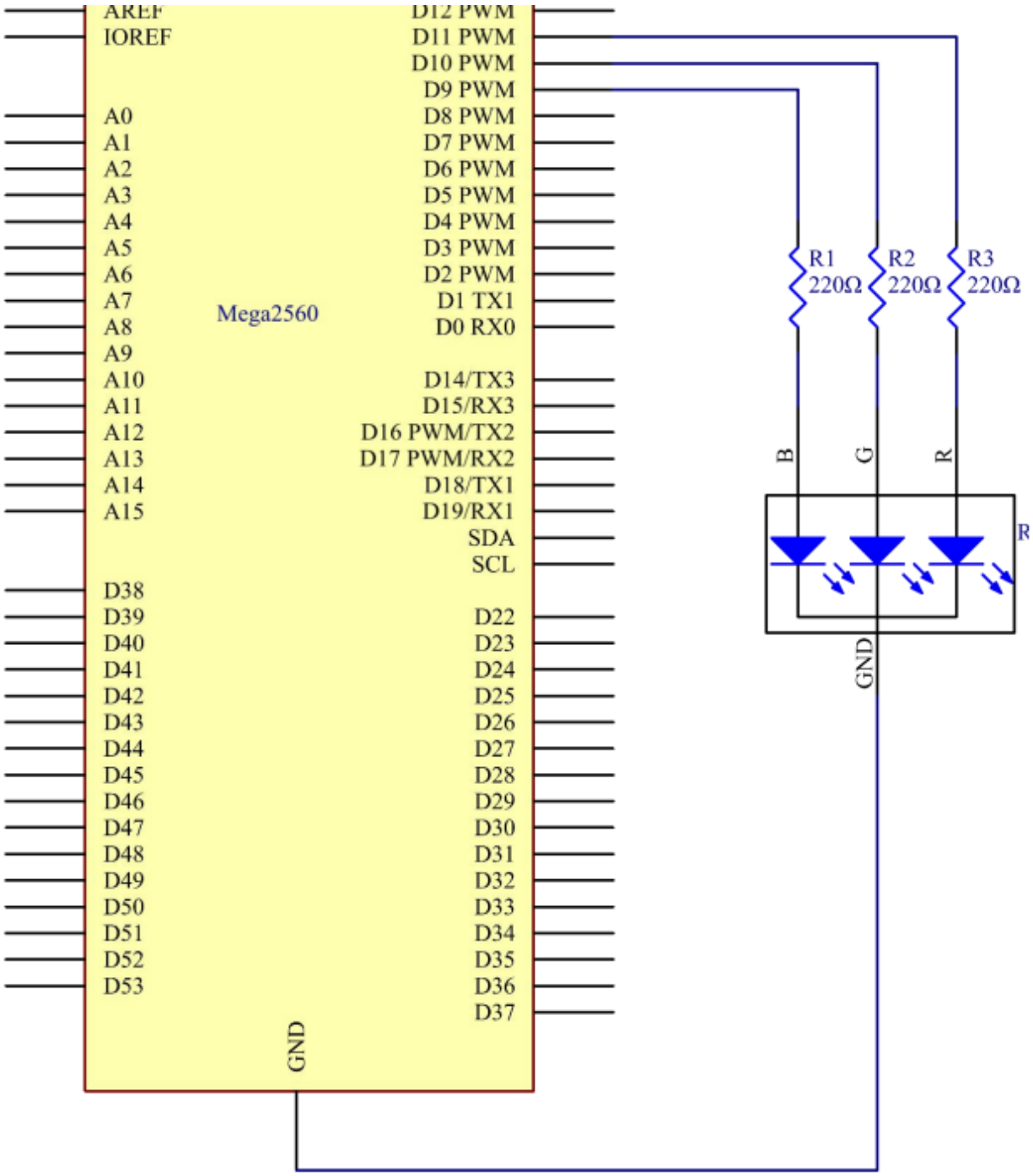
对 analogWrite() 的调用范围为 0 - 255，这样 `analogWrite(255)` 需要一个 100% 占空比（始终打开），`analogWrite(127)` 是 50% 占空比（一半时间)。

你会发现 PWM 值越小，转换成电压后的值越小。然后 LED 相应地变暗。因此，我们可以通过控制 PWM 值来控制 LED 的亮度。

7.7.4 原理图

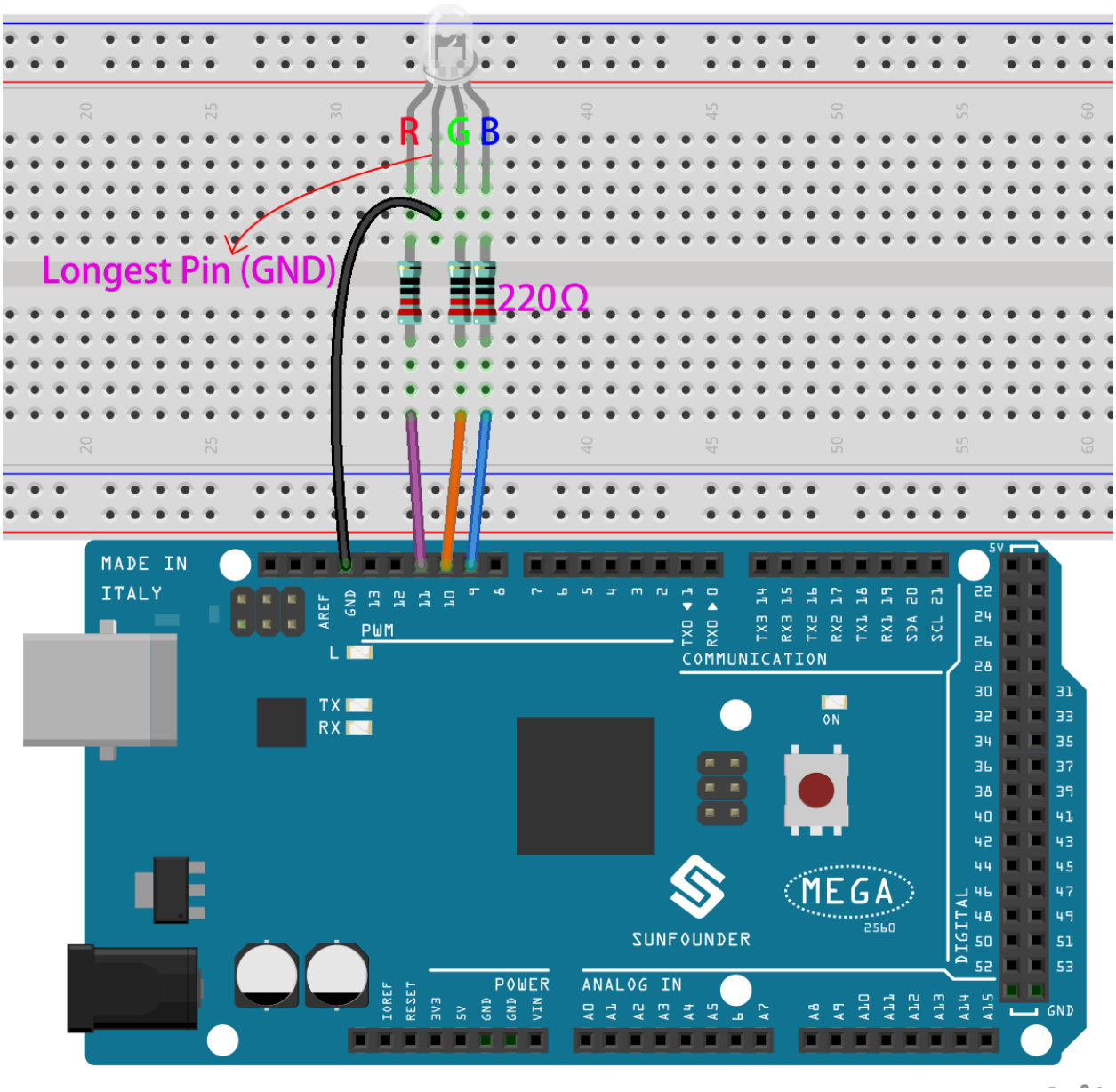
在 Mega2560 板上，2 到 13 和 44 到 46 是 PWM 引脚，通过 `analogWrite()` 函数提供 8 位 PWM 输出。你可以连接这些引脚中的任何一个。这里我们将 0 到 255 之间的值输入到 RGB LED 的三个引脚，使其显示不同的颜色。R、G、B 引脚接限流电阻后，分别接 9、10、11 脚。LED 最长的引脚（GND）连接到控制板的 GND。当三个引脚被赋予不同的 PWM 值时，RGB LED 将显示不同的颜色。

原理图如下所示：



7.7.5 实验步骤

第 1 步：搭建电路。

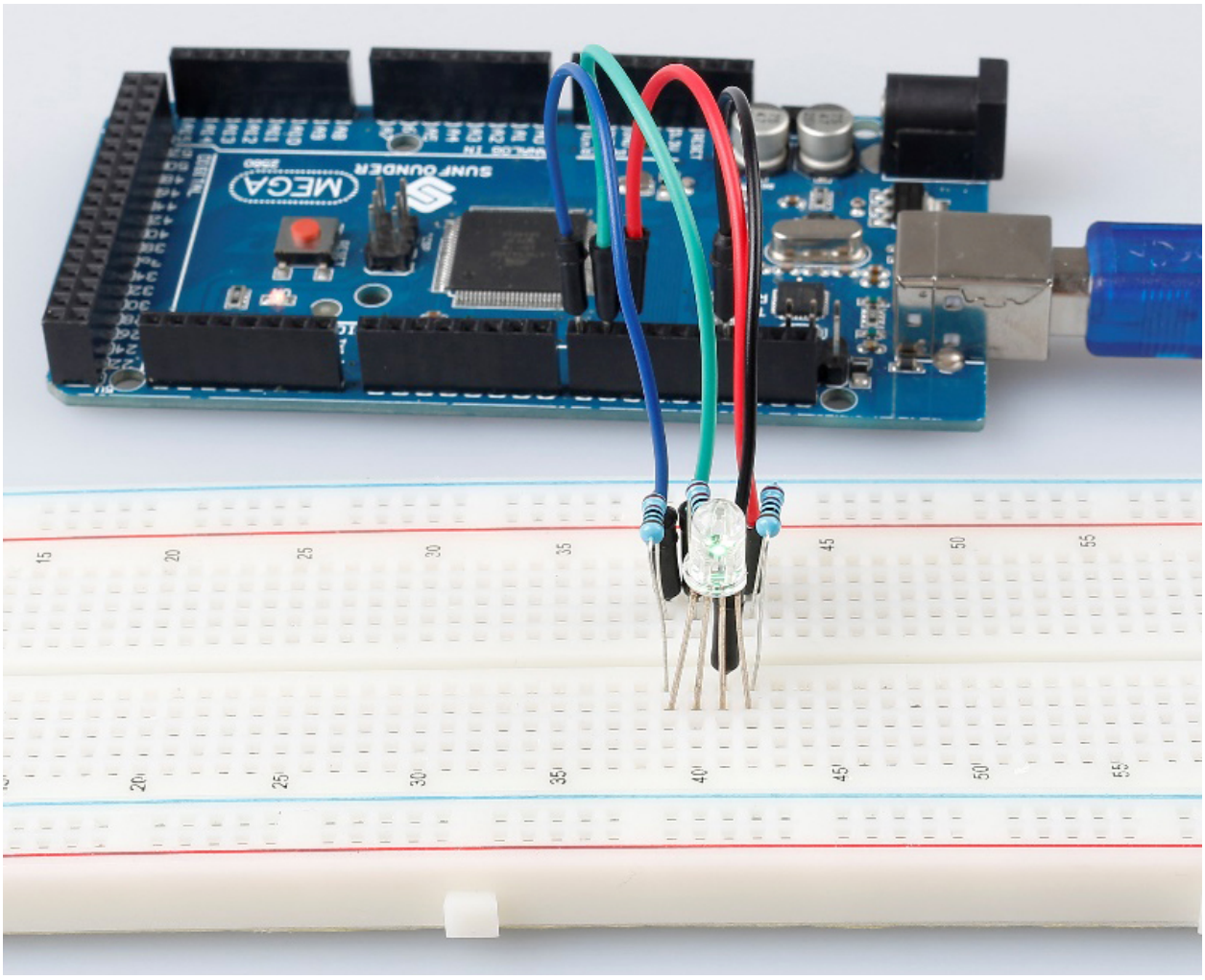


第 2 步：打开代码文件 `Lesson_7_RGB_LED.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

在这里你应该看到 RGB LED 首先循环闪烁红色、绿色和蓝色，然后是红色、橙色、黄色、绿色、蓝色、靛蓝色和紫色。



7.7.6 代码

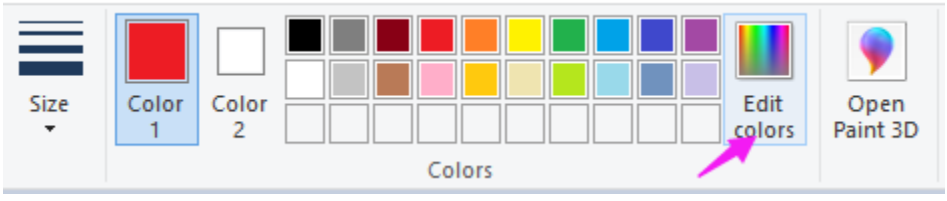
7.7.7 代码分析

设置颜色

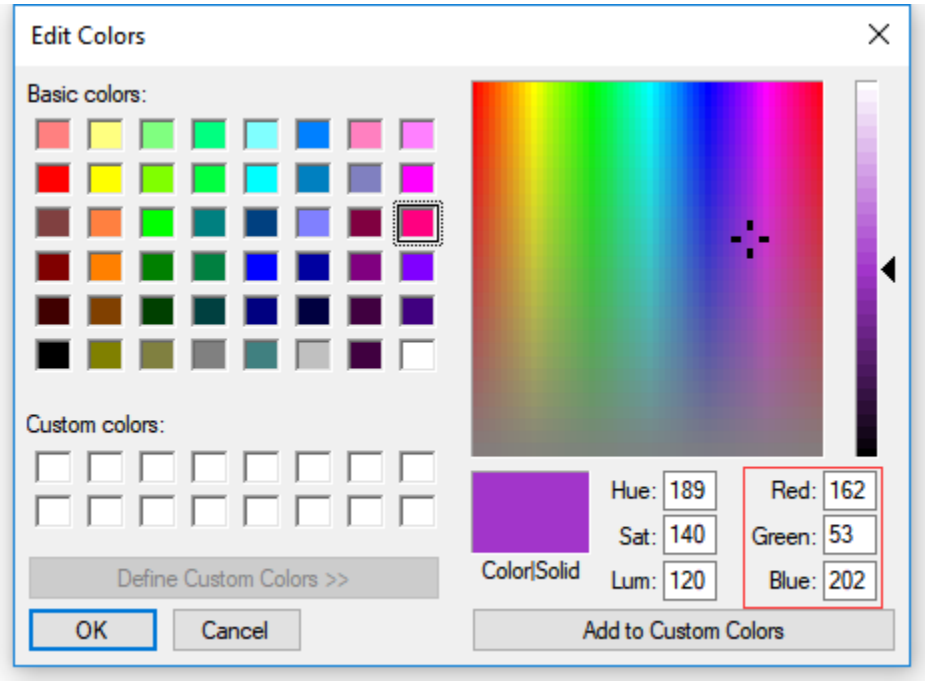
这里使用 `color()` 函数来设置 RGB LED 的颜色。在代码中，它被设置为闪烁 7 种不同的颜色。

你可以使用计算机上的绘图工具获取 RGB 值。

1. 打开计算机上的绘画工具，然后单击编辑颜色。



2. 选择一种颜色，即可看到该颜色的 RGB 值。在代码中填写它们。




```
void loop()  // run over and over again
{
    // Basic colors:
    color(255, 0, 0); // turn the RGB LED red
    delay(1000); // delay for 1 second
    color(0,255, 0); // turn the RGB LED green
    delay(1000); // delay for 1 second
    color(0, 0, 255); // turn the RGB LED blue
    delay(1000); // delay for 1 second
    // Example blended colors:
    color(255,0,252); // turn the RGB LED red
    delay(1000); // delay for 1 second
    color(237,109,0); // turn the RGB LED orange
    delay(1000); // delay for 1 second
    color(255,215,0); // turn the RGB LED yellow
    delay(1000); // delay for 1 second
    color(34,139,34); // turn the RGB LED green
    delay(1000); // delay for 1 second
    color(0,112,255); // turn the RGB LED blue
    delay(1000); // delay for 1 second
    color(0,46,90); // turn the RGB LED  indigo
    delay(1000); // delay for 1 second
    color(128,0,128); // turn the RGB LED purple
    delay(1000); // delay for 1 second
}
```

color() 函数

```
void color (unsigned char red, unsigned char green, unsigned char blue)// the color␣
↪generating function
{
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
```

(续下页)

(接上页)

```
}
```

定义三个无符号字符变量，红色、绿色和蓝色。将它们的值写入 `redPin`、`greenPin` 和 `bluePin`。例如，颜色（128,0,128）是写 128 到 redPin，0 至 greenPin 和 128 至 bluePin。然后结果是 LED 闪烁紫色。

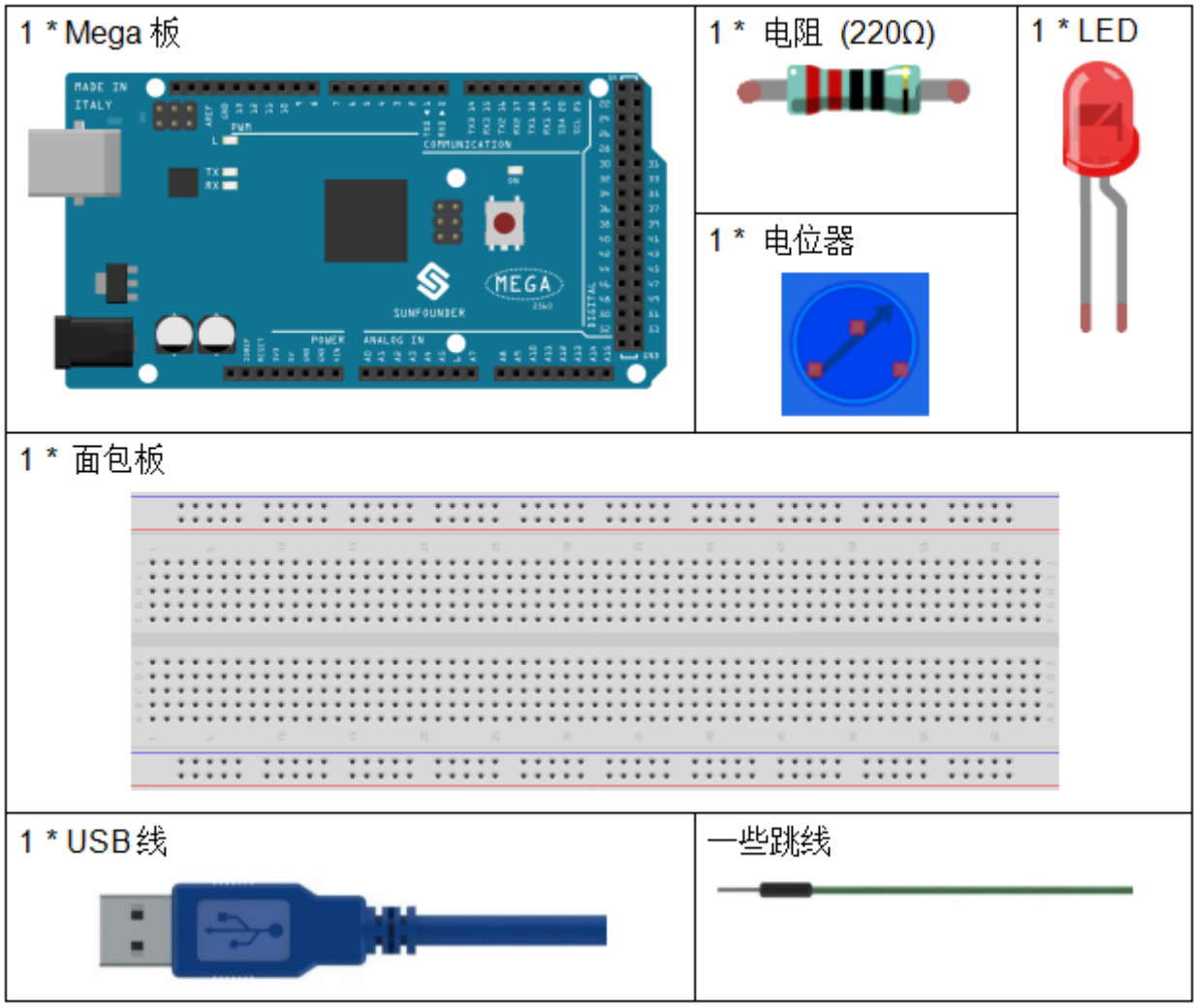
- `analogWrite()`：将模拟值（PWM 波）写入引脚。它与模拟引脚无关，仅适用于 PWM 引脚。在调用 `analogWrite()` 之前，你不需要调用 `pinMode()` 将引脚设置为输出。

7.8 第 8 课电位器

7.8.1 介绍

在本课中，让我们看看如何通过电位器改变 LED 的亮度，并在串口监视器中接收电位器的数据以查看其值变化。

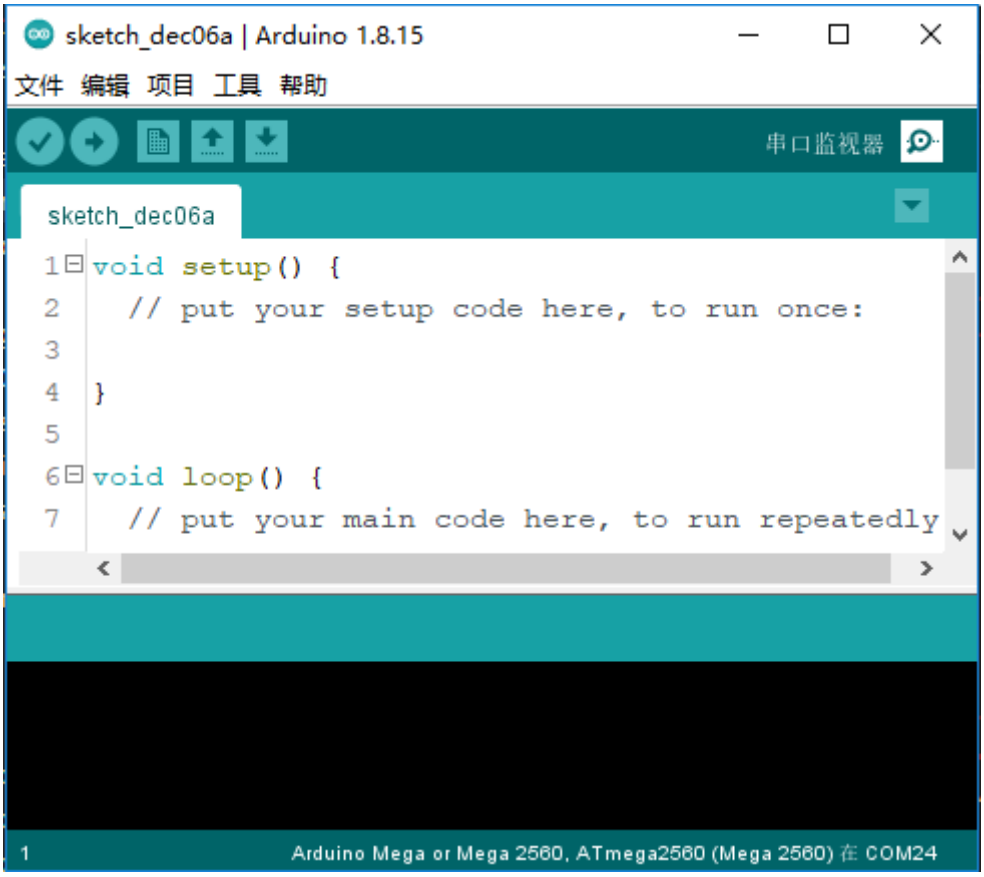
7.8.2 所需器件



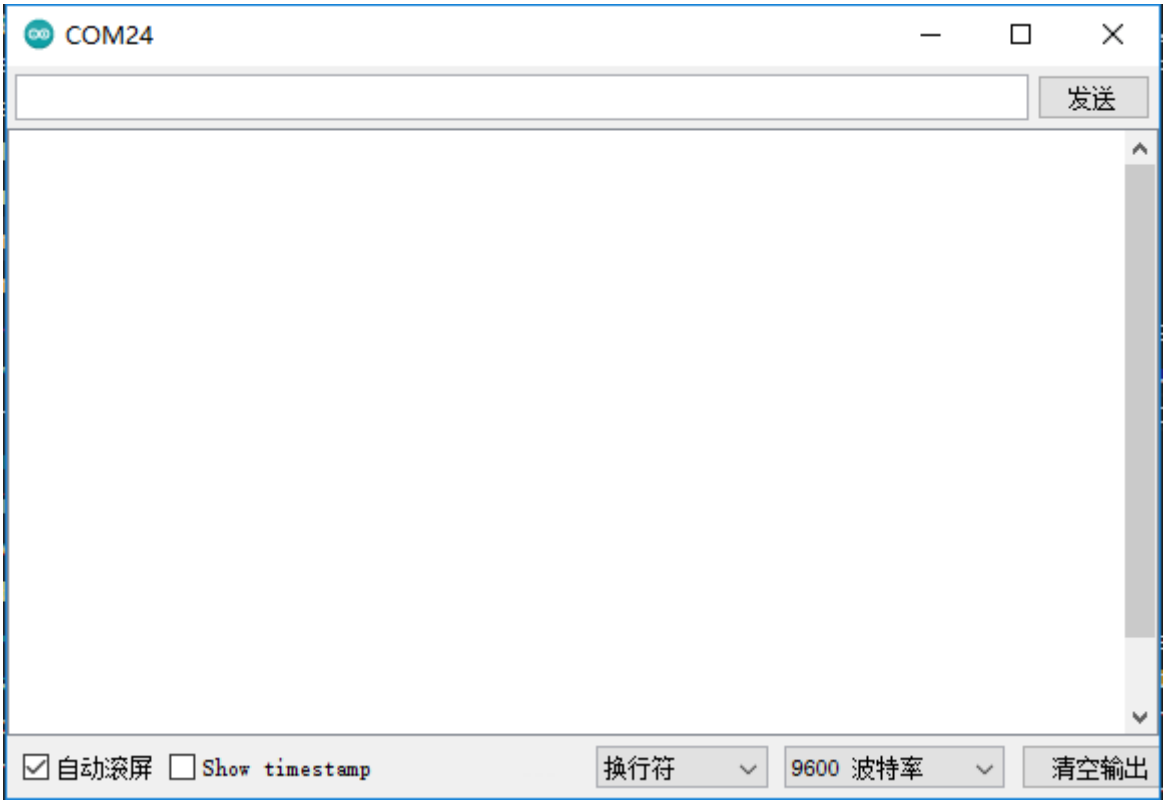
- *SunFounder Mega 板*
- 面包板
- 跳线
- *LED 发光二极管*
- 电阻
- 电位器

7.8.3 串行监视器

串行监视器用于控制板和计算机或其他设备之间的通信，可以用来发送和接收数据。它是 Arduino 环境中的内置软件，你可以点击右上角的按钮打开它。

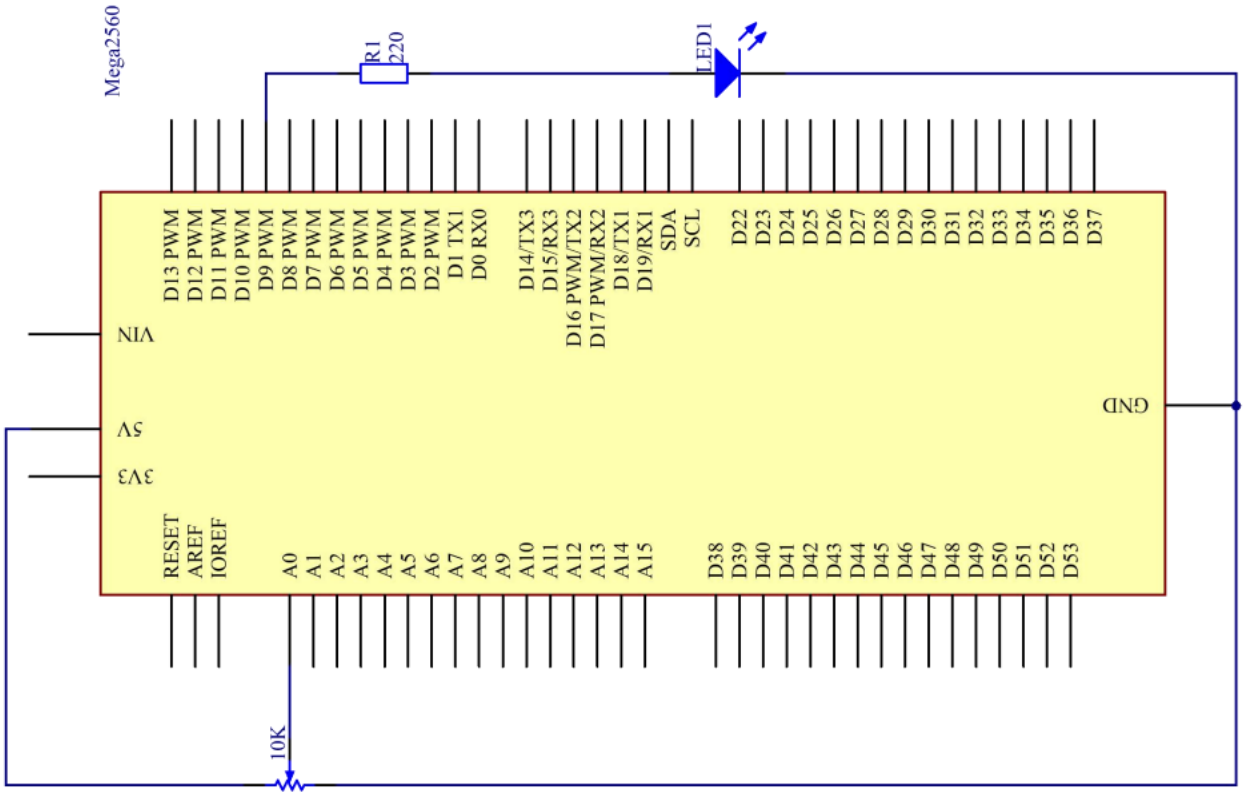


在这里，串行监视器用作计算机和控制板之间通信的中转站。首先，计算机将数据传输到 **串口监视器**，然后由控制板读取数据。最后在控制板进行相关操作。点击右上角的图标，会弹出一个窗口，如下图：



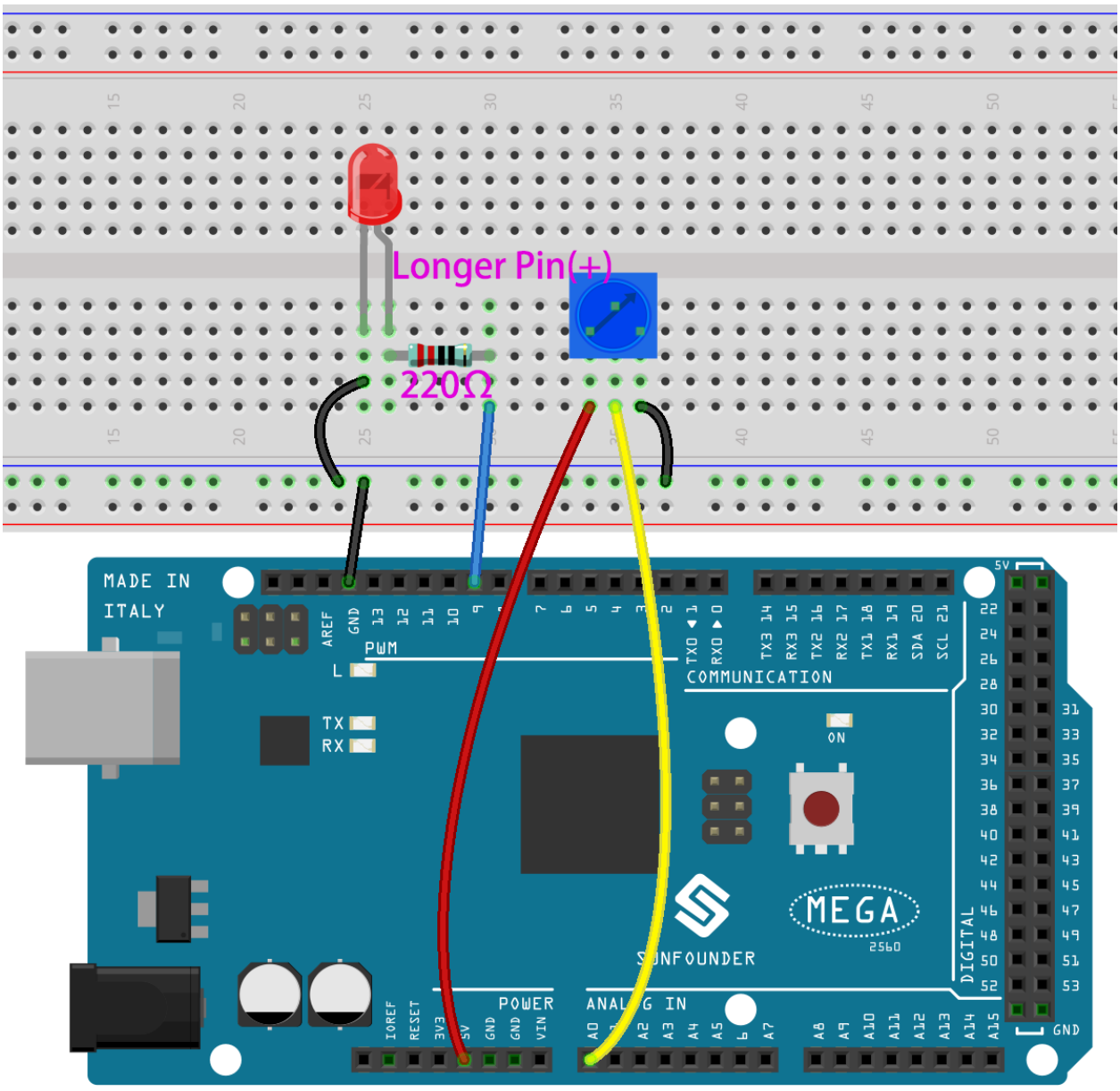
7.8.4 原理图

在这个实验中，电位器用作分压器，这意味着将设备连接到它的所有三个引脚。将电位器的中间引脚连接到引脚 A0，另外两个引脚分别连接到 5V 和 GND。因此，电位器的电压为 0-5V。旋转电位器的旋钮，A0 脚的电压会发生变化。然后使用控制板中的 AD 转换器将该电压转换为数字值 (0-1024)。通过编程，我们可以使用转换后的数字值来控制控制板上 LED 的亮度。



7.8.5 实验步骤

第 1 步：搭建电路。



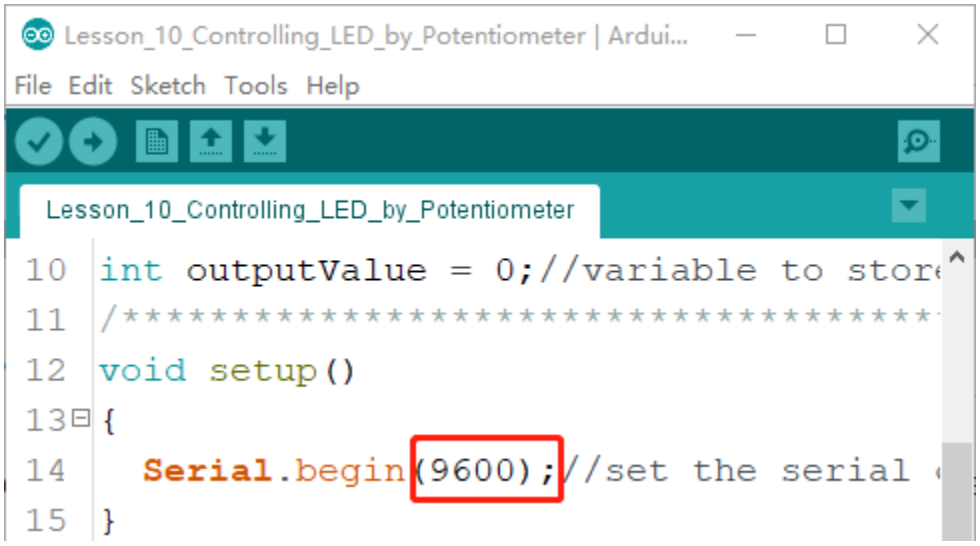
第 2 步：打开代码文件 `Lesson_8_Potentiometer.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

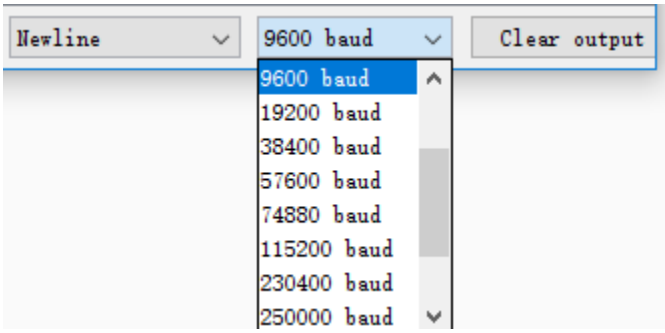
第 5 步：打开串口监视器。

找到 `Serial.begin()`，看看设置了什么波特率，这里是 9600。然后点击右上角的图标打开串口监视器。



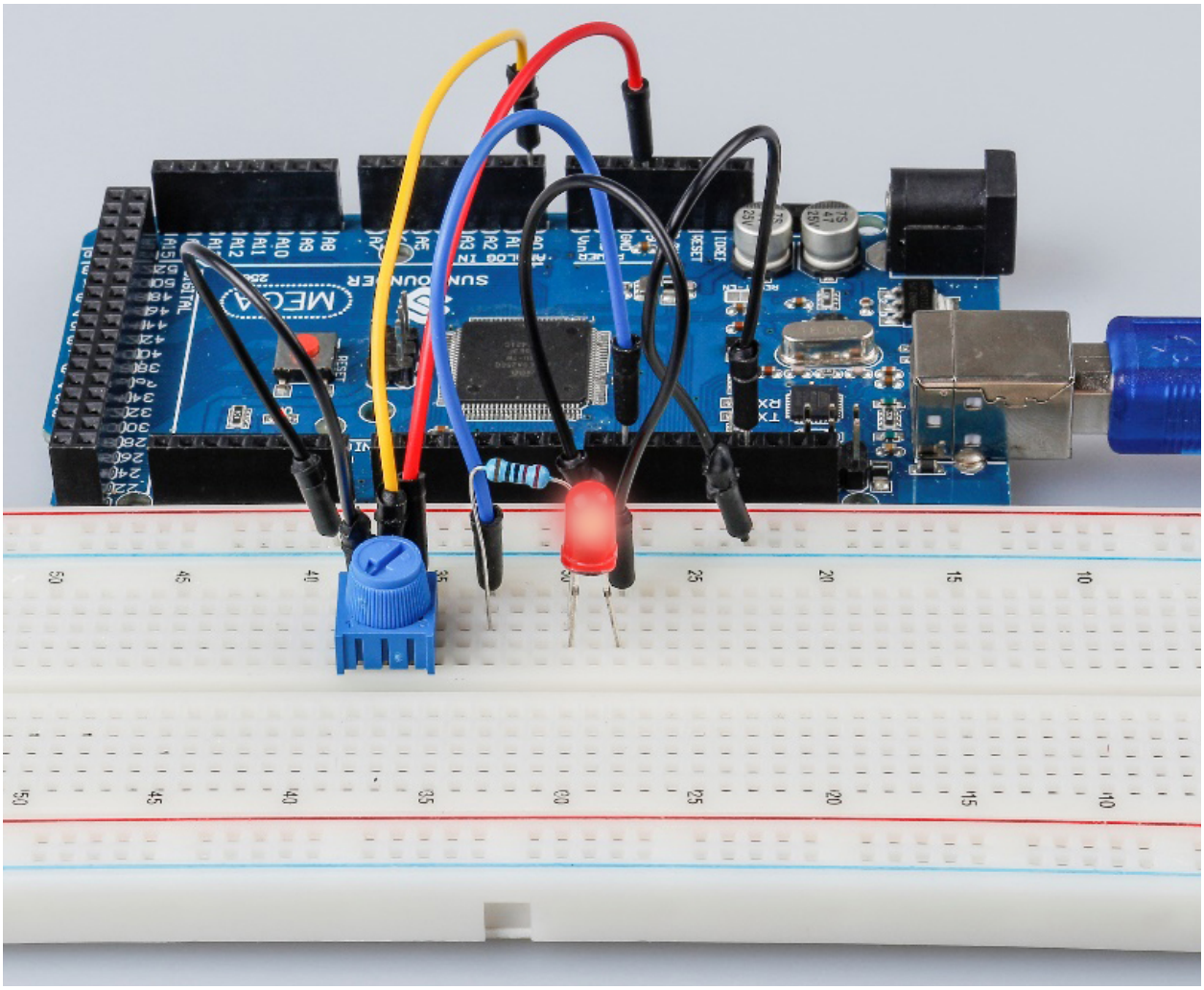
第 6 步：设置波特率为 9600。

串口监视器的默认波特率为 9600，如果代码也设置为 9600，则无需更改波特率栏。



旋转电位器的轴，你应该看到 LED 的亮度发生变化。

如果要检查相应值的变化，请打开串行监视器，窗口中的数据将随着你旋转电位器旋钮而变化。



7.8.6 代码

7.8.7 代码分析

从 A0 读取值

```
inputValue = analogRead(analogPin);//read the value from the potentiometer
```

这一行是将 A0 读取的值存储在之前定义的 `inputValue` 中。

`analogRead()` 从指定的模拟引脚读取值。这意味着它会将 0 到 5 伏之间的输入电压映射为 0 到 1023 之间的整数值。

在串行监视器上打印值

```
Serial.print("Input: "); //print "Input"
Serial.println(inputValue); //print inputValue
```

- `Serial.print()`：将数据作为人类可读的 ASCII 文本打印到串口。这个命令可以有多种形式。数字被打印为每个数字的 ASCII 字符。浮点数同样被打印为 ASCII 数字，默认为两位小数。字节以单个字符的形式发送。字符和字符串按原样发送。

- `Serial.println()`：与 `Serial.print()` 相同，但它后面有一个回车字符（ASCII 13，或'r'）和一个换行字符（ASCII 10，或'n'）。

将值映射

```
outputValue = map(inputValue, 0, 1023, 0, 255); //Convert from 0-1023 proportional to␣
↪the number of a number of from 0 to 255
```

- `map(value, fromLow, fromHigh, toLow, toHigh)` 函数是将数字从一个范围重新映射到另一个范围。也就是说，值 fromLow 将被映射到了 tolow，值 fromHigh 到 toHigh，值之间以值之间，等等。

由于 `ledPin` 的范围是 0-255，我们需要将 0-1023 映射到 0-255。

以同样的方式在串口监视器中显示输出值。如果你对 `map()` 函数不是很清楚，你可以观察串口监视器中的数据并进行分析。

```
Serial.print("Output: "); //print "Output"

Serial.println(outputValue); //print outputValue
```

将电位器的值写到 LED 上

```
analogWrite(ledPin, outputValue); //turn the LED on depending on the output value
```

将输出值写入 `ledPin`，你将看到 LED 的亮度随着你旋转电位器旋钮而变化。

- `analogWrite()`：将模拟值（PWM 波）写入引脚。它与模拟引脚无关，仅适用于 PWM 引脚。在调用 `analog Write()` 之前，你不需要调用 `pinMode()` 将引脚设置为输出。

7.8.8 实验总结

这个实验也可以随意改成其他的。例如，使用电位器来控制 LED 闪烁的时间间隔。就是利用从电位器读取的数值进行延时，如下图。试试!

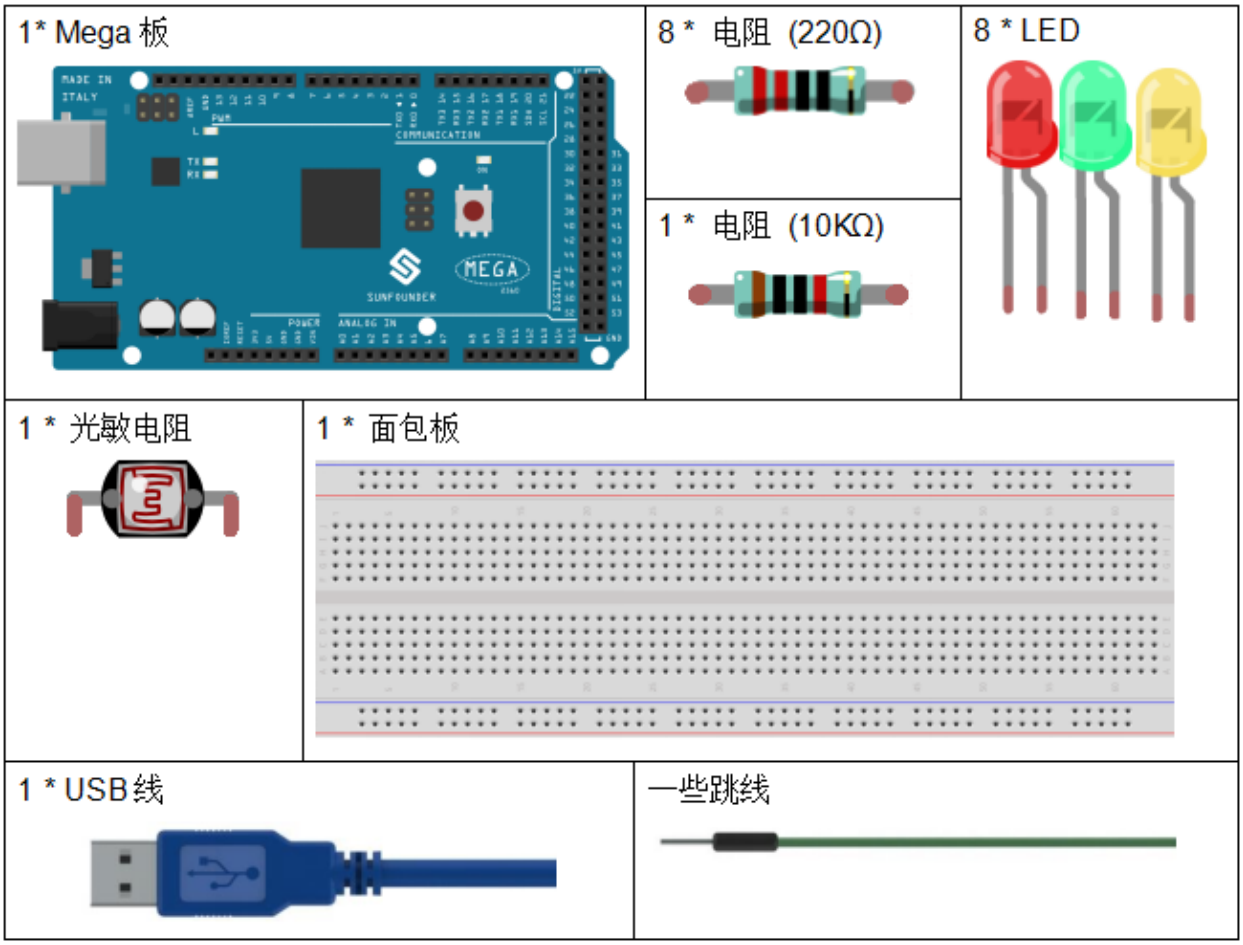
```
inputValue = analogRead(analogPin);
digitalWrite(ledPin, HIGH);
delay(inputValue);
digitalWrite(ledPin, LOW);
delay(inputValue);
```

7.9 第 9 课光敏电阻

7.9.1 介绍

在本课中，你将学习如何使用光敏电阻测量光强度。光敏电阻的电阻值随着入射光强度的变化而变化。如果光照强度变高，阻值减少；如果光强度变暗，阻值就会增加。

7.9.2 所需器件

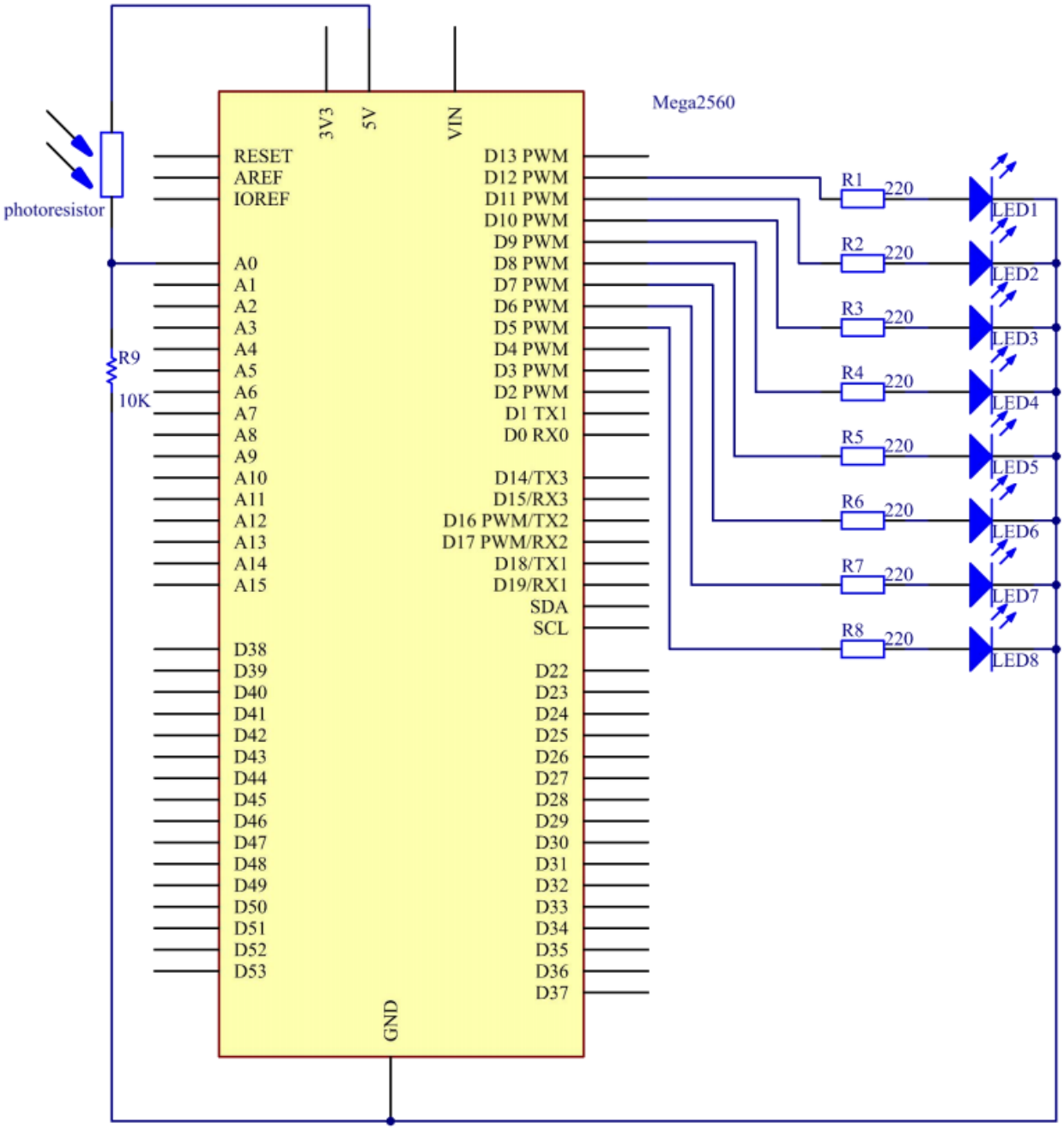


- *SunFounder Mega 板*
- 面包板
- 跳线
- *LED 发光二极管*
- 电阻
- 光敏电阻

7.9.3 原理图

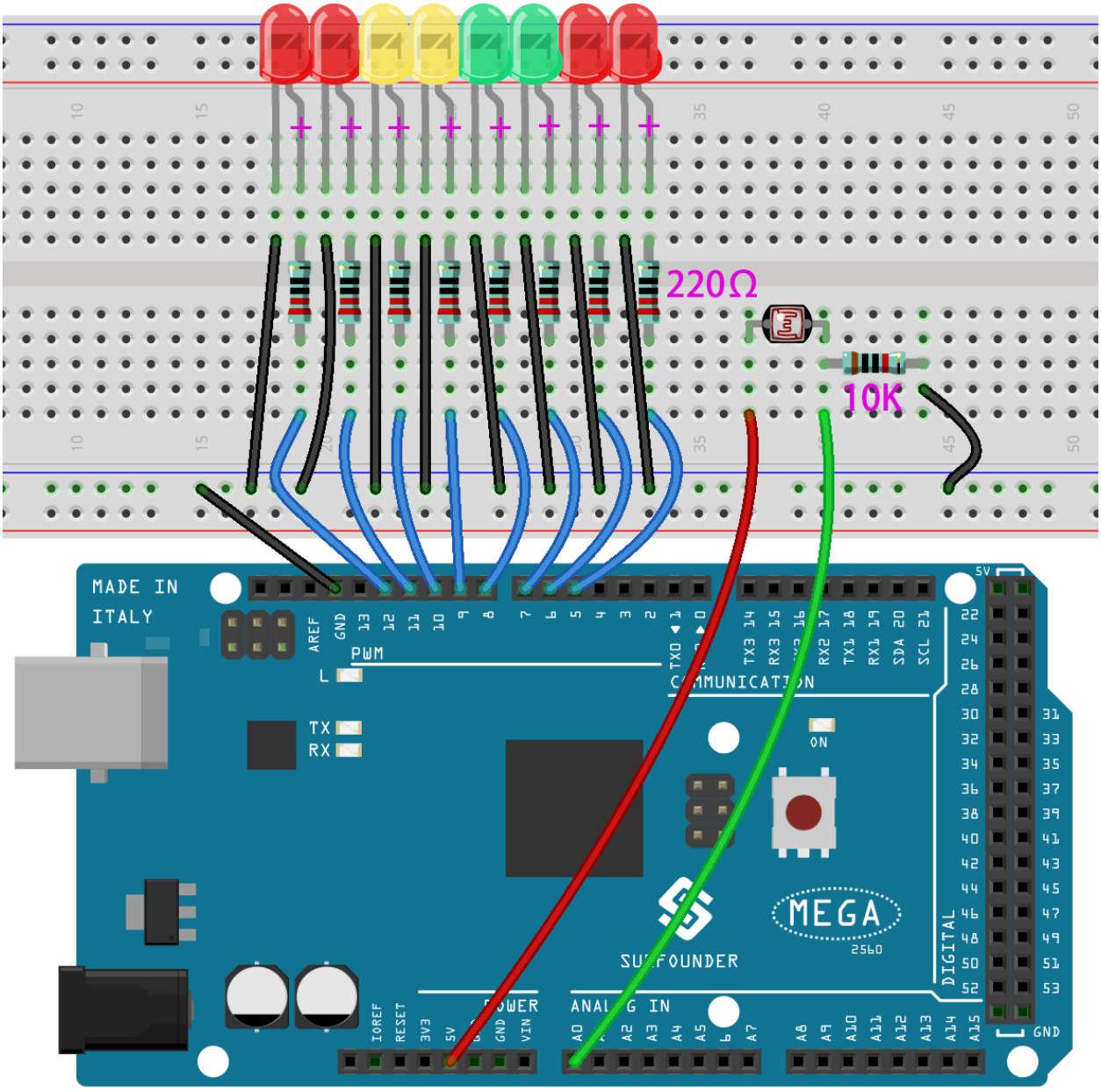
在这个实验中，我们将用 8 个 LED 灯来显示光的强度。光强度越高，越多的 LED 灯会亮起来。当光强度足够高时，所有的 LED 灯都会亮起来。当没有光，所有的 LED 灯都会熄灭。

原理图如下所示：


Mega2560
3V3
5V
VIN
GND
RESET
AREF
IOREF
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
D38
D39
D40
D41
D42
D43
D44
D45
D46
D47
D48
D49
D50
D51
D52
D53
D13 PWM
D12 PWM
D11 PWM
D10 PWM
D9 PWM
D8 PWM
D7 PWM
D6 PWM
D5 PWM
D4 PWM
D3 PWM
D2 PWM
D1 TX1
D0 RX0
D14/TX3
D15/RX3
D16 PWM/TX2
D17 PWM/RX2
D18/TX1
D19/RX1
SDA
SCL
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
D32
D33
D34
D35
D36
D37
photoresistor
R9
10K
R1
220
LED1
R2
220
LED2
R3
220
LED3
R4
220
LED4
R5
220
LED5
R6
220
LED6
R7
220
LED7
R8
220
LED8

7.9.4 实验步骤

第 1 步：搭建电路。

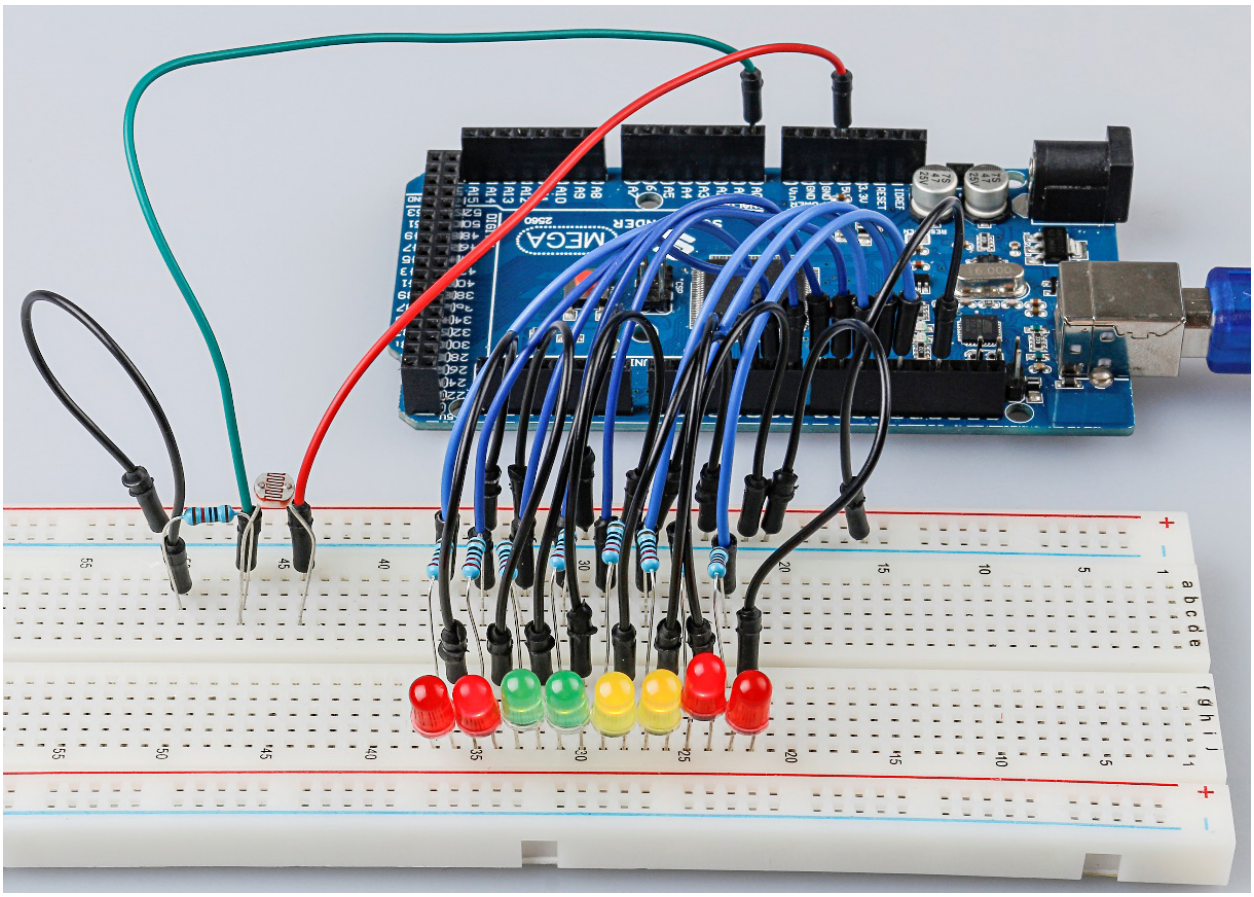


第 2 步：打开代码文件 `Lesson_9_Photoresistor.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，用一些光照在光敏电阻上，你会看到几个 LED 灯亮起来。照更多的光，你会看到更多的 LED 灯亮起来。当你把它放在一个黑暗的环境中，所有的 LED 都会熄灭。



7.9.5 代码

7.9.6 代码分析

设置变量

```
const int NbrLEDs = 8; // 8 leds
const int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9}; // 8 leds attach to pin 5-12␣
↪respectively
const int photocellPin = A0; // photoresistor attach to A0
int sensorValue = 0; // value read from the sensor
int ledLevel = 0; // sensor value converted into LED 'bars'
```

8 个 LED 被连接到 5 引脚-12 引脚，在这段代码中，使用一个数组来存储这些引脚，`ledPins[0]` 等于 5，`ledPins[1]` 等于 6，以此类推。

设置 8 个引脚为输出

```
for (int led = 0; led < NbrLEDs; led++)
{
    pinMode(ledPins[led], OUTPUT); // make all the LED pins outputs
}
```

使用 `for()` 语句将 8 个引脚依次设置为输出。依次为 OUTPUT。

读取光敏电阻的模拟值。

```
sensorValue = analogRead(photocellPin); // read the value of A0
```

读取 photocellPin (A0 引脚) 的值并存储到变量 sensorValue 中。

- analogRead()：从指定的模拟引脚读取数值。Arduino 板包含一个多通道、10 位的模拟数字转换器。这意味着它将映射出 0 到工作电压（5V 或 3.3V）之间的输入电压。电压（5V 或 3.3V）之间的输入电压映射为 0 至 1023 之间的整数值。

```
Serial.print("SensorValue: ");
Serial.println(sensorValue); // Print the analog value of the photoresistor
```

使用 Serial.print() 函数来打印光敏电阻的模拟值，你将在串口监视器中看到它们。

- Serial.print()：将数据作为人类可读的 ASCII 文本打印到串口。这个命令可以有多种形式。数字被打印为每个数字的 ASCII 字符。浮点数同样被打印为 ASCII 数字，默认为两位小数。字节以单个字符的形式发送。字符和字符串按原样发送。
- Serial.println()：与 Serial.print() 相同，但它后面有一个回车字符（ASCII 13，或'r'）和一个换行字符（ASCII 10，或'n'）。

将模拟值映射到 8 个 LED 上

```
ledLevel = map(sensorValue, 0, 1023, 0, NbrLEDs); // map to the number of LEDs
Serial.print("ledLevel: ");
Serial.println(ledLevel);
```

这个 map() 函数是用来将 0-1023 映射到 0-NbrLEDs(8)。

(1023-0)/(8-0)=127.875

0-12	128-2	2 56-38	384-	5 12-63	640-7	7 68-89	896 -
7.875	55.75	3.625	511.5	9.375	67.25	5.125	1023
0	1	2	3	4	5	6	7

如果 sensorValue 等于 560，则 ledLevel 为 4.

- map(value, fromLow, fromHigh, toLow, toHigh) 函数是将数字从一个范围重新映射到另一个范围。也就是说，值 fromLow 将被映射到了 tolow，值 fromHigh 到 toHigh，值之间以值之间，等等。

点亮 LED 灯

```
for (int led = 0; led < NbrLEDs; led++)
{
    if (led <= ledLevel ) //When led is smaller than ledLevel, run the following code.
    {
        digitalWrite(ledPins[led], HIGH); // turn on pins less than the level
    }
    else
    {
        digitalWrite(ledPins[led], LOW); // turn off pins higher than
    }
}
```

点亮相应的 LED。例如，当 ledLevel 为 4 时，点亮 ledPins[0] 到 ledPins[4]，熄灭 ledPins[5] 到 ledPins[7]。

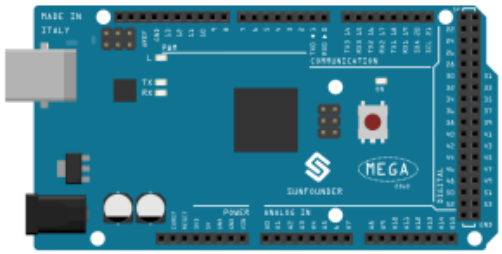
7.10 第 10 课舵机

7.10.1 介绍

伺服是一种只能旋转 180 度的减速电机。它是通过从你的电路板发送电脉冲来控制的。这些脉冲告诉伺服它应该移动到什么位置。

舵机有三根线：棕色线为 GND，红色线为 VCC，橙色线为信号线。

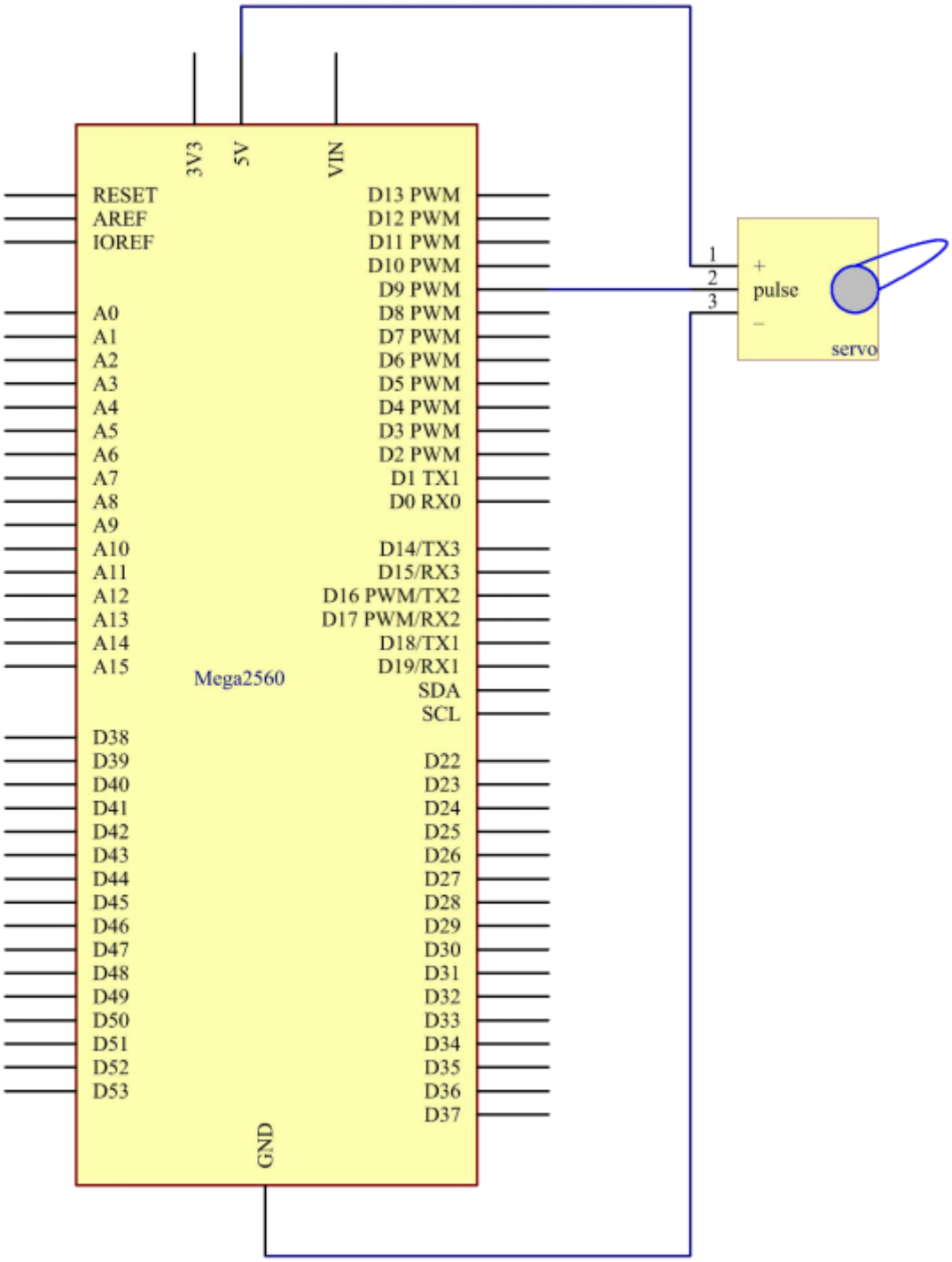
7.10.2 所需器件

<table>
<tr>
<td>1 * Mega 板
  </td>
<td>1 * USB 线
 
一些跳线(公-母)
  </td>
<td>1 * 舵机
  </td>
</tr>
</table>

- *SunFounder Mega 板*
- 面包板
- 跳线
- 舵机

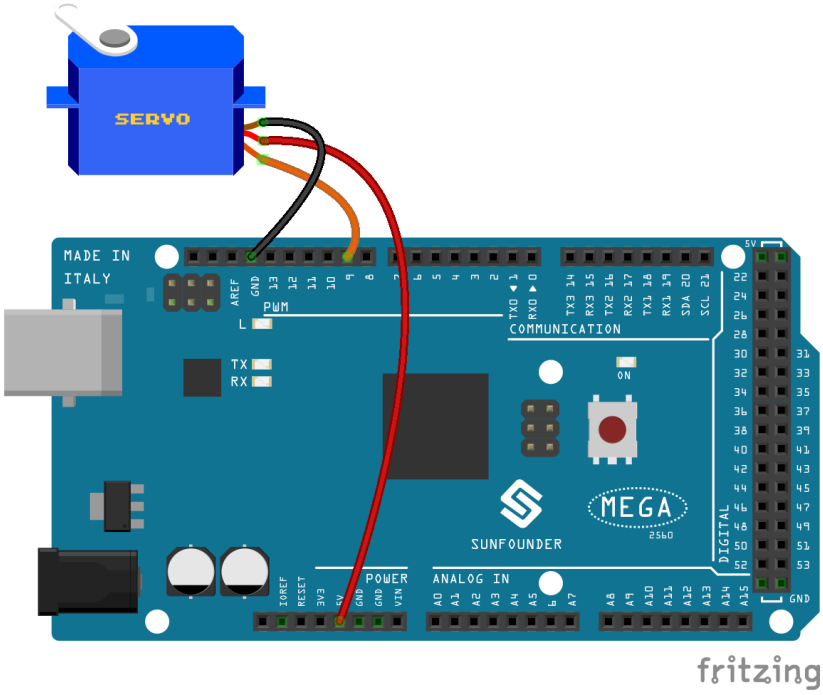
7.10.3 原理图

原理图如下所示：



7.10.4 实验步骤

第 1 步：搭建电路。(棕色连接到 GND，橙色连接到 9 引脚，红色连接到 5V)。

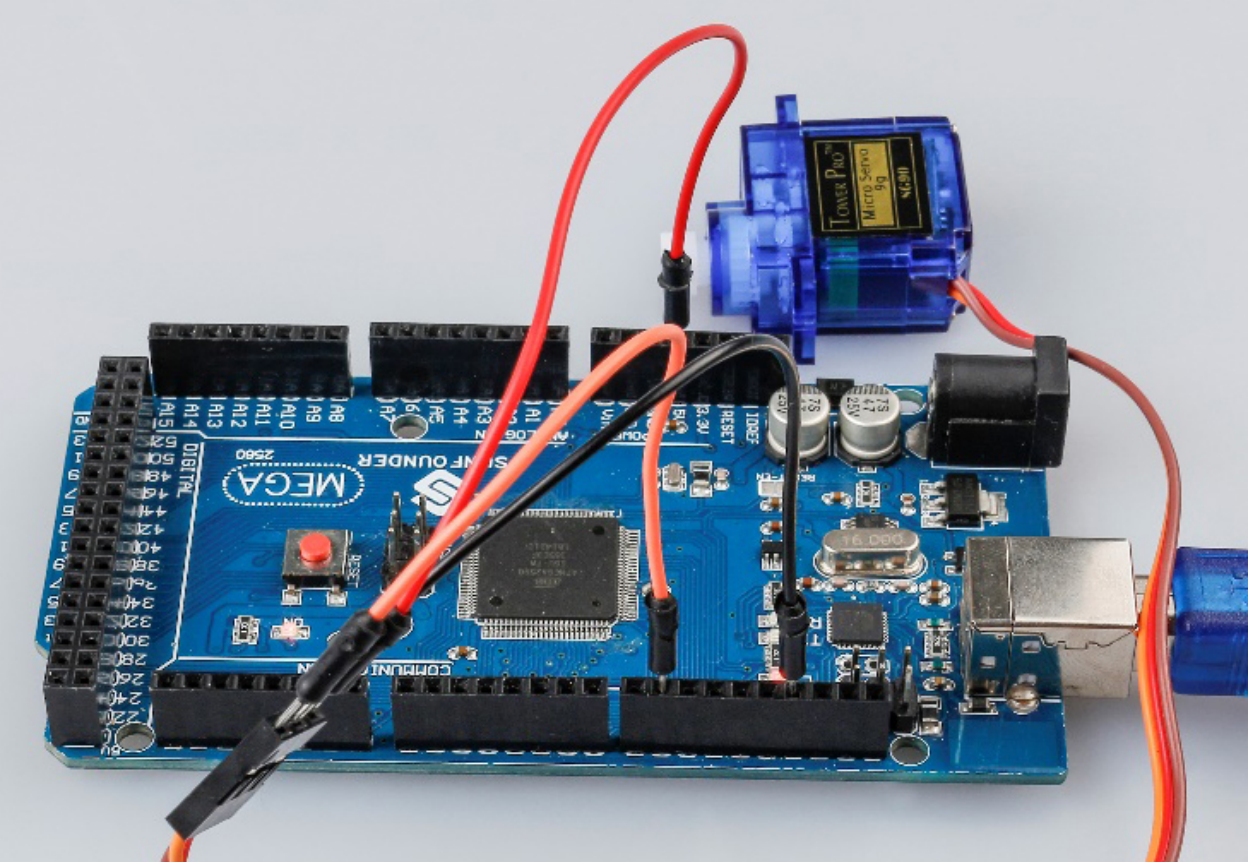


第 2 步：打开代码文件 `Lesson_10_Servo.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你可以看到舵机的摇臂旋转并停止在 90 度（每次 15 度）。然后它向相反的方向旋转。



7.10.5 代码

7.10.6 代码分析

添加一个库

```
#include <Servo.h>
Servo myservo; //create servo object to control a servo
```

导入 Servo.h 文件之后，你就可以在这个文件中的函数。Servo 是 Arduino IDE 中的内置库。你可以在安装路径，默认是 C:Program FilesArduinolibraries 下找到 Servo 文件夹。

初始化舵机

```
void setup()
{
    myservo.attach(9); //attachs the servo on pin 9 to servo object
    myservo.write(0); //back to 0 degrees
    delay(1000); //wait for a second
}
```

- myservo.attach()：用来初始化舵机，并设置它的信号引脚。
- myservo.write()：将一个值写入舵机，相应地控制它的轴。在一个标准的舵机上，这将设置轴的角度 (度)，将轴移到那个方向。这里让伺服机首先保持在 0 角度。

让舵机转动

```
void loop()
{
    for (int i = 0; i <= 180; i++)
    {
        myservo.write(i); //write the i angle to the servo
        delay(15); //delay 15ms
    }
    for (int i = 180; i >= 0; i--)
    {
        myservo.write(i); //write the i angle to the servo
        delay(15); //delay 15ms
    }
}
```

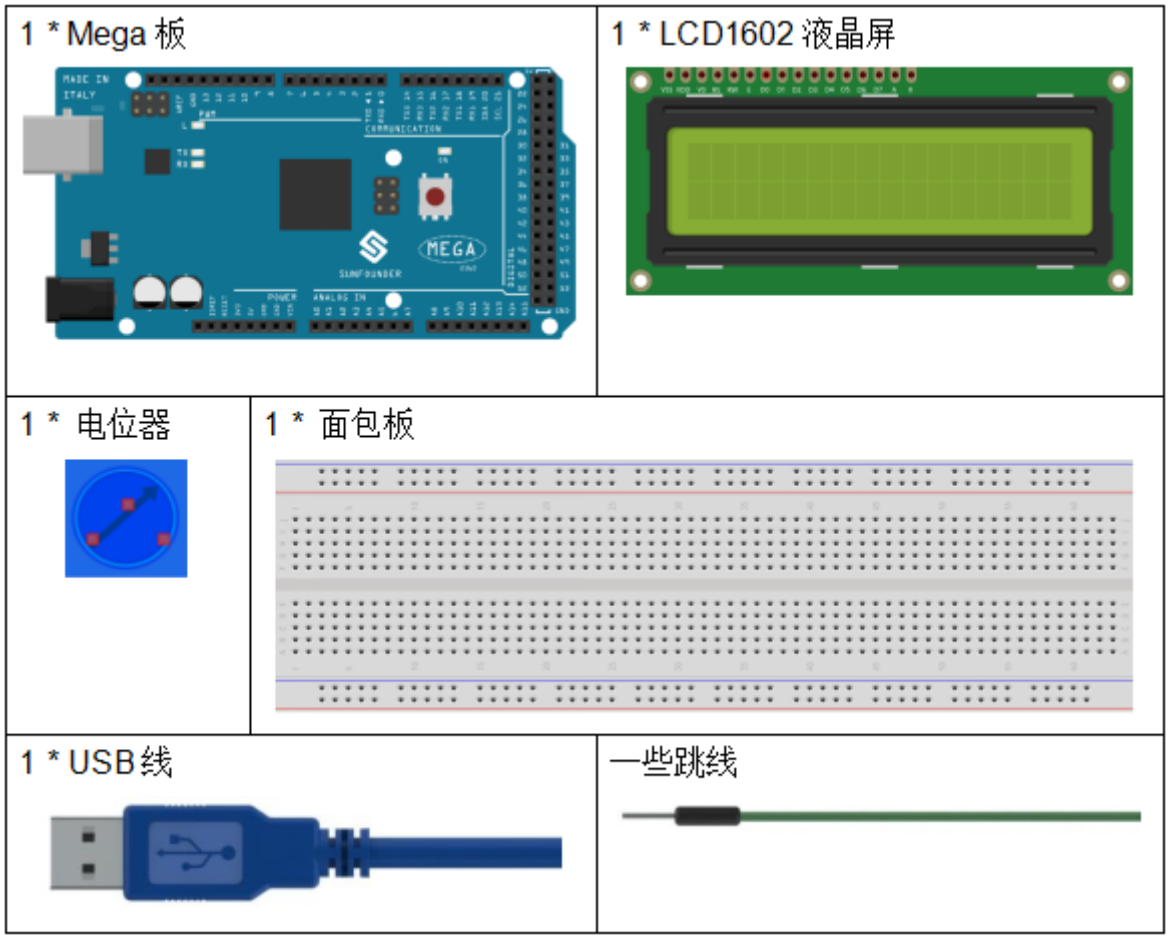
用 2 个 `for()` 语句将 0-180 写入舵机，这样就可以看到舵机从 0 转到 180 角，然后转回 0。

7.11 第 11 课 LCD1602

7.11.1 介绍

在本课中，我们将学习如何使用 LCD1602 来显示字符和字符串。LCD1602，即 1602 字符型液晶显示器，是一种显示字母、数字、字符等的点阵模块。它由 5x7 或 5x11 点阵位置组成；每个位置可以显示一个字符。两个字符之间有一个点间距，行之间有一个空格，从而将字符和行分开。数字 1602 表示在显示屏上可以显示 2 行，每行 16 个字符。现在让我们学习下如何使用它。

7.11.2 所需器件

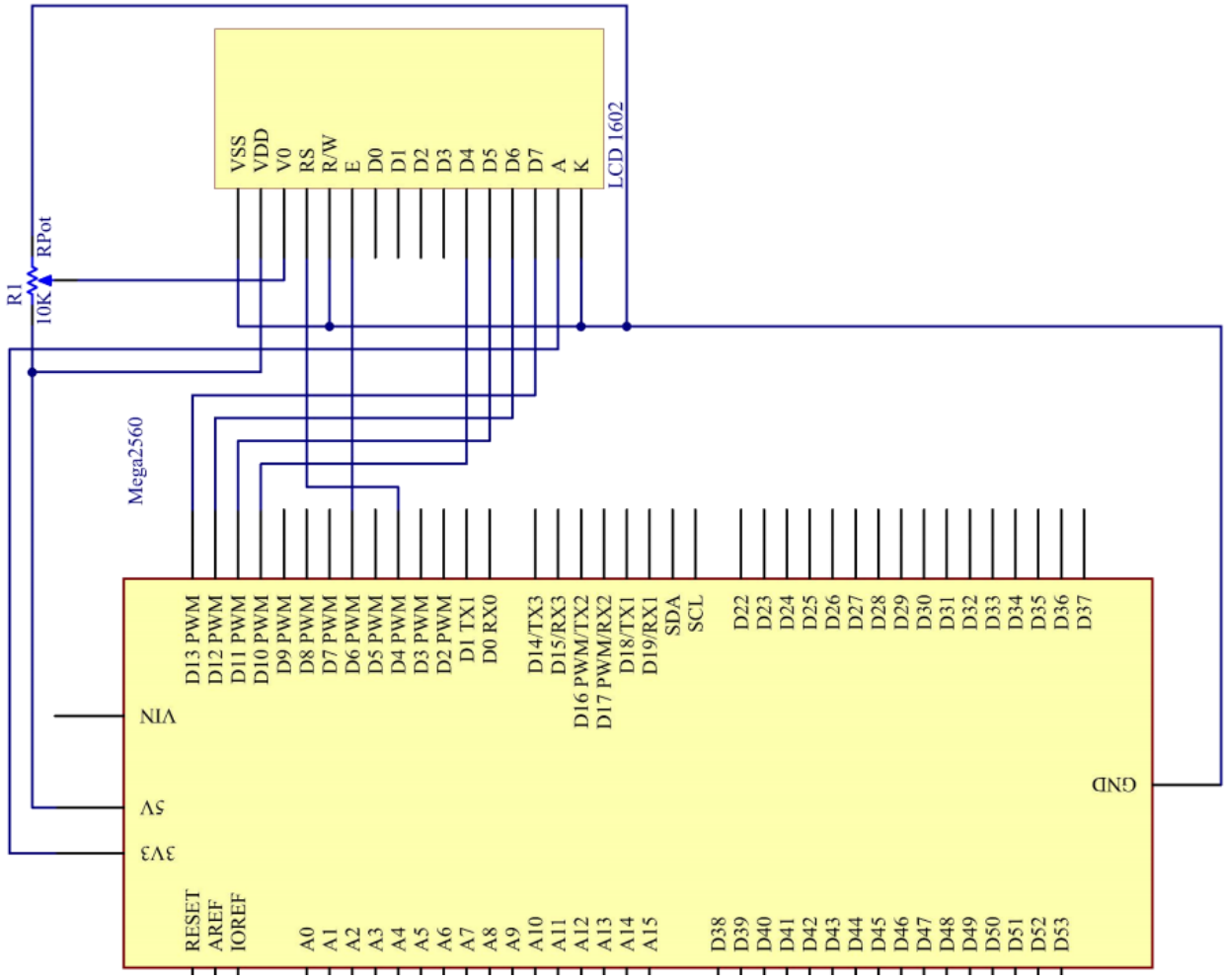


- *SunFounder Mega 板*
- *面包板*
- *跳线*
- *LCD1602 液晶显示屏*
- *电位器*

7.11.3 原理图

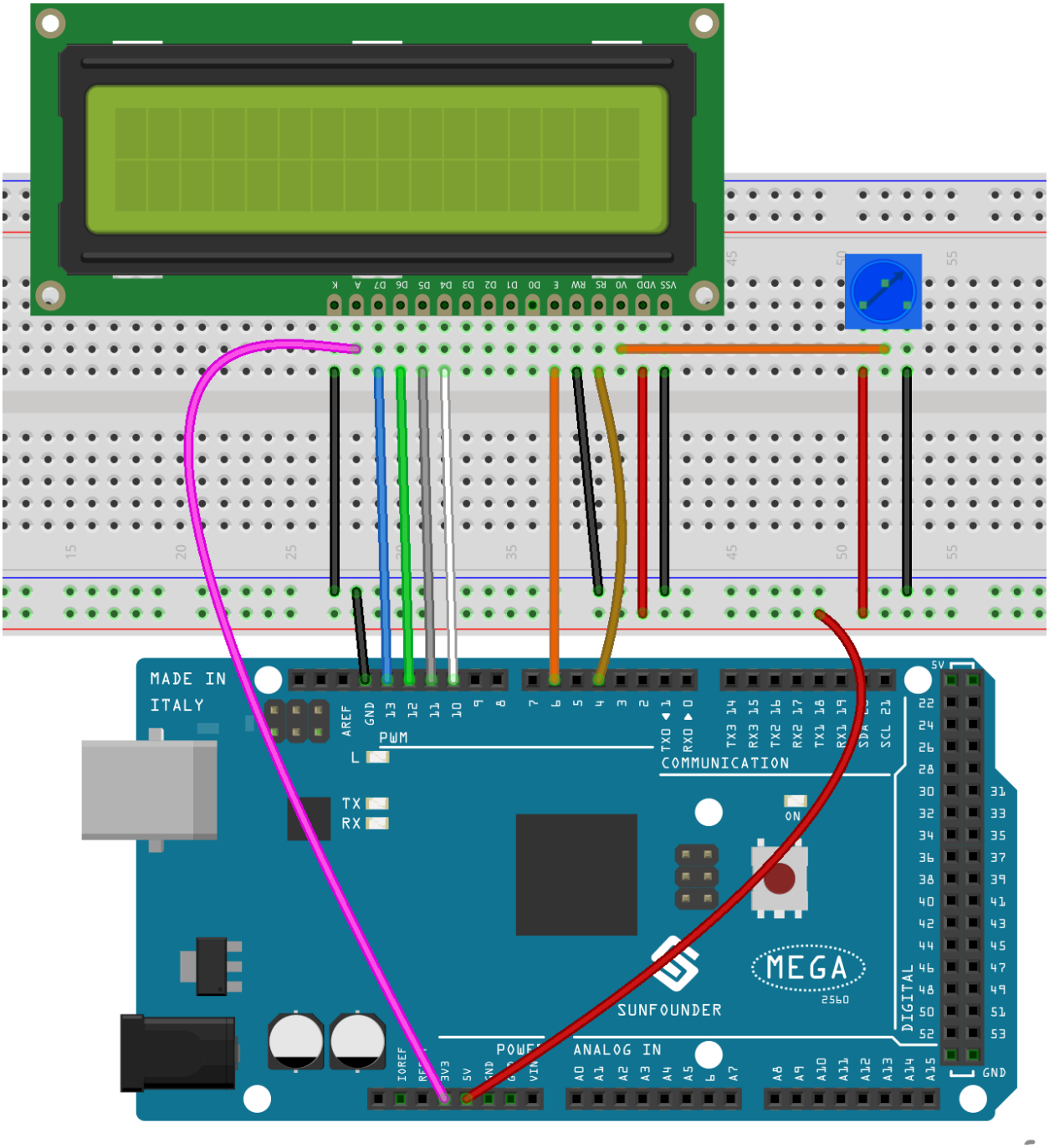
- 连接 **K** 到 GND 和 **A** 至 3.3 V，然后将 LCD1602 的背光将被打开。
- 将 **VSS** 连接到 GND。
- 将 **VO** 连接到电位器的中间引脚 - 你可以使用它来调整屏幕显示的对比度。
- 连接 RS 到 D4 和 R / W 连接到 GND，该装置则可以写入字符的 LCD1602。
- 将 **E** 接 6 引脚，LCD1602 上显示的字符由 **D4-D7** 控制。

原理图如下所示：



7.11.4 实验步骤

第 1 步：搭建电路。（请仔细按下图接线，以免液晶屏无法正常工作。）



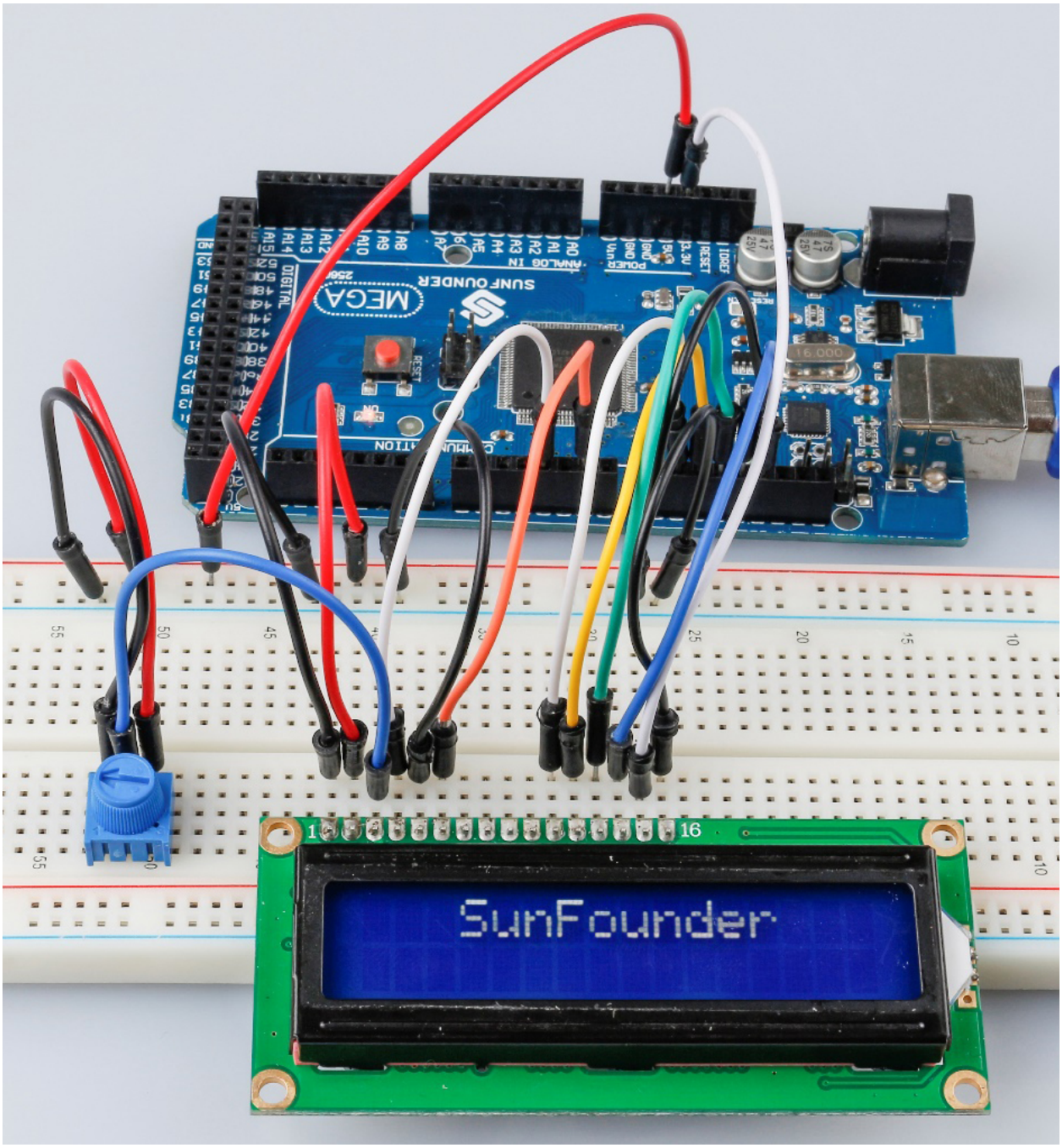
第 2 步：打开代码文件 `Lesson_11_LCD1602.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

备注： 代码上传完成之后，若不显示或者显示不清晰，可旋转电位器来调整对比度（亮/暗显示比例）。

你现在应该看到字符 `SunFounder` 和 `hello, world` 在 LCD1602 上滚动。


SunFounder

7.11.5 代码

7.11.6 代码分析

导入一个库

```
#include <LiquidCrystal.h>// include the library code
```

由于包含了 LiquidCrystal.h 文件，你可以在以后调用该文件中的函数。

LiquidCrystal 是 Arduino IDE 中的一个内置库。你可以在安装路径, 默认是 C:\Program Files\Arduino\libraries 下找到 LiquidCrystal 文件夹。

在 examples 文件夹包含的是相关的示例代码。src 文件夹包含了库的主要部分：LiquidCrystal.cpp``(执行文件，包括函数实现、变量定义等）和 ``LiquidCrystal.h`` (头文件，包括函数声明、宏定义、结构定义等)。如果你想探索某个函数是如何实现的，你可以在 ``LiquidCrystal.cpp 文件中查找。

需显示的字符串

```
char array1[]=" SunFounder "; //the string to print on the LCD
char array2[]="hello, world! "; //the string to print on the LCD
```

这是两个字符型数组：arry1[] 和 array2[]。引号 "xxx " 中的内容是它们的元素，总共包括 26 个字符（空格算在内）。array1[0] 代表数组中的第一个元素，是一个空格，array1[2] 意味着第二个元素 S，以此类推。所以 array1[25] 是最后一个元素（这里也是一个空格）。

定义 LCD1602 的引脚

```
LiquidCrystal lcd(4, 6, 10, 11, 12, 13);
```

定义一个 LiquidCrystal 类型的变量 lcd。这里用 lcd 来表示下面代码中的 LiquidCrystal。

- LiquidCrysral() 函数的基本格式是：LiquidCrystal (rs, enable, d4, d5, d6, d7)。你可以查看 LiquidCrystal.cpp 文件了解详情。

所以这一行定义了 RS 脚与 4 脚相连，enable 脚与 6 脚相连，d4-d7 分别与 10-13 脚相连。

初始化 LCD1602

```
lcd.begin(16, 2); // set up the LCD's number of columns and rows: begin(col,row) is␣
↪to set the display of LCD. Here set as 16 x 2.
```

设置光标的位置

```
lcd.setCursor(15,0); // set the cursor to column 15, line 0
```

- setCursor(col,row) 用来设置光标的位置，即开始显示字符的地方。这里把它设置为 15 列（第 16 列），0 行（第 1 行）。

LCD1602 显示字符

```
for ( int positionCounter1 = 0; positionCounter1 < 26; positionCounter1++)
{
    lcd.scrollDisplayLeft(); //Scrolls the contents of the display one space to the␣
↪left.
    lcd.print(array1[positionCounter1]); // Print a message to the LCD.
    delay(tim); //wait for 250 microseconds
}
```

当 positionCounter1 = 0 时，与 positionCounter1 < 26 一致。positionCounter1 加 1，通过 lcd.scrollDisplayLeft() 向左移动一位。通过 lcd.print(array1[positionCounter1]) 使 LCD 显示 array1[0]，并延迟 tim ms（250ms）。循环 26 次后，array1[] 中的所有元素都被显示。

```
lcd.clear(); //Clears the LCD screen.
```

用 lcd.clear() 清除屏幕，这样它就不会影响下次的显示了。

```
lcd.setCursor(15,1); // set the cursor to column 15, line 1 // Set the cursor at Col.␣
↪15 Line 1, where the characters will start to show.
for (int positionCounter2 = 0; positionCounter2 < 26; positionCounter2++)
{
    lcd.scrollDisplayLeft(); //Scrolls the contents of the display one space to the␣
↪left.
    lcd.print(array2[positionCounter2]); // Print a message to the LCD.
    delay(tim); //wait for 250 microseconds
}
```

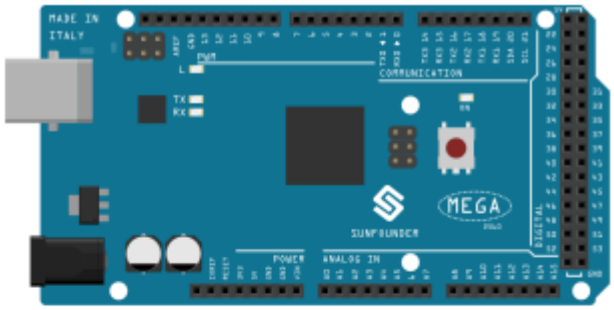
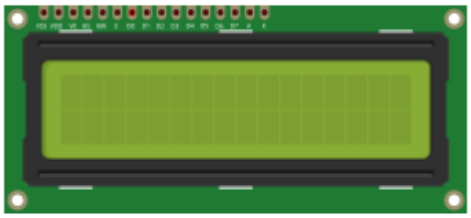
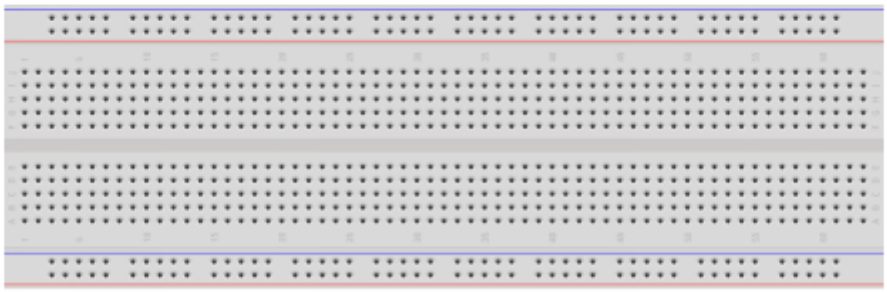
同样地，代码是在 LCD 上显示 array2[] 中的元素。因此，你会看到 SunFounder 在 LCD 的第一行向左移动直到消失。然后在第二行显示 hello, world !，同时也向左滚动直到消失。

7.12 第 12 课热敏电阻

7.12.1 介绍

到目前为止，我们已经学习了很多设备。要做更多的东西，你需要掌握更多的知识。今天我们要认识一个热敏电阻。它类似于光敏电阻，能够根据外部变化来改变其电阻。与光敏电阻不同，热敏电阻的电阻值随外部环境温度的变化而显着变化。

7.12.2 所需器件

1 * Mega 板 	1 * 热敏电阻 	1 * 电位器 
1 * LCD1602 液晶屏 	1 * 电阻(10KΩ) 	
1 * 面包板 		
1 * USB线 	一些跳线 	

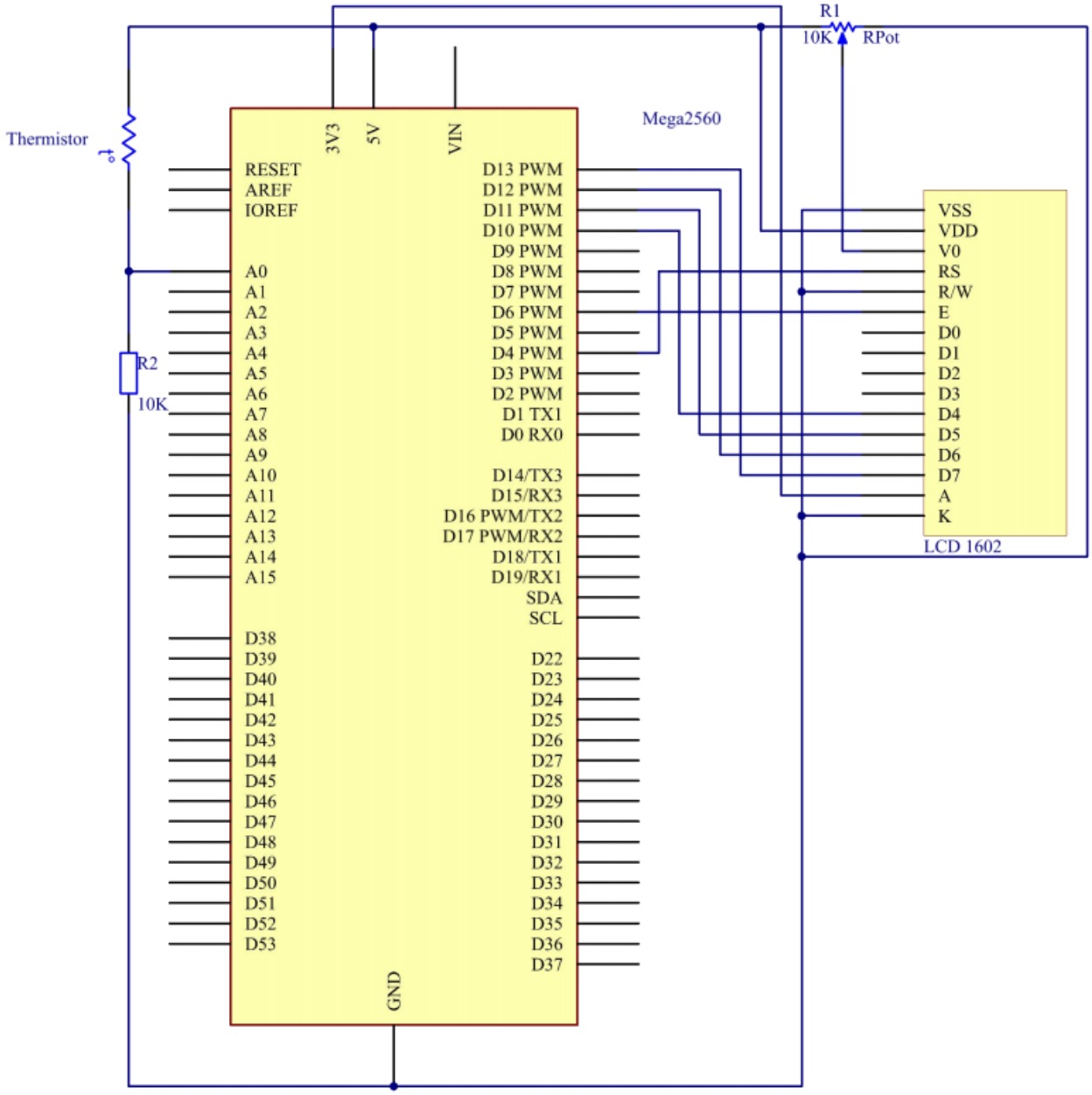
- *SunFounder Mega 板*
- 面包板
- 跳线
- 电阻
- 电位器
- 热敏电阻
- *LCD1602 液晶显示屏*

7.12.3 原理图

热敏电阻是一种敏感元件，它有两种类型：负温度系数（NTC）和正温度系数（PTC），也有 NTC 和 PTC。其电阻随温度显着变化。当 NTC 的电阻降低时，PTC 热敏电阻的电阻随着温度的升高而增加。

在这个实验中，我们使用一个 NTC。

原理图如下所示：



其原理是 NTC 热敏电阻的阻值随着外界环境的温差而变化。它检测环境的实时温度。当温度升高时，热 敏电阻的阻值减小，A0 脚电压相应升高。然后电压数据由 A/D 适配器转换为数字量。然后通过编程输出摄氏和华氏温度，然后显示在 LCD1602 上。

在这个实验中，使用了一个热敏电阻和一个 10k 的上拉电阻。每个热敏电阻都有一个正常的电阻。这里是 10k ohm，这是在 25 摄氏度下测量的。

这是电阻和温度变化之间的关系：

R T = R N exp B(1/TK –1/TN)

- RT 是 NTC 热敏电阻在温度为 TK 时的阻值。
- RN 是 NTC 热敏电阻在额定温度为 TN 的阻值。
- TK 是开尔文温度，单位是 K。
- TN 是额定开尔文温度；单位也是 K。
- 并且 β，这里是 NTC 热敏电阻的材料常数，也称为热敏指数。
- exp 是指数的缩写，是一个以 e 为基数的指数，它是一个自然数，大约等于 2.7。

请注意，此关系是一个经验公式。只有当温度和电阻在有效范围内时才准确。

由于 TK = T+273，T 为摄氏温度，电阻与温度变化的关系可转化为：

R =R o exp B[1/(T+273) –1/(To+273)]

B 是 beta 的缩写，是一个常数。这里是 4090。R o 是 10k 欧姆，T o 是 25 摄氏度。数据可以在热敏电阻的数据表中找到。同样，上述关系可以转化为一个来评估温度：

T= B/[ln(R/ 10) + (B/ 298)] –273（所以 ln 在这里表示自然对数，以 e 为底的对数）

如果我们使用固定电阻为 10k ohms 的电阻，我们可以用这个公式计算模拟输入引脚 A0 的电压：

V =10k x 5/(R+10K)

所以，可以形成这种关系：

R = (5 x 10k /V) - 10k

A0 的电压通过 A/D 适配器转换成数字 a。

a=V x (1024/5)

V=a/205

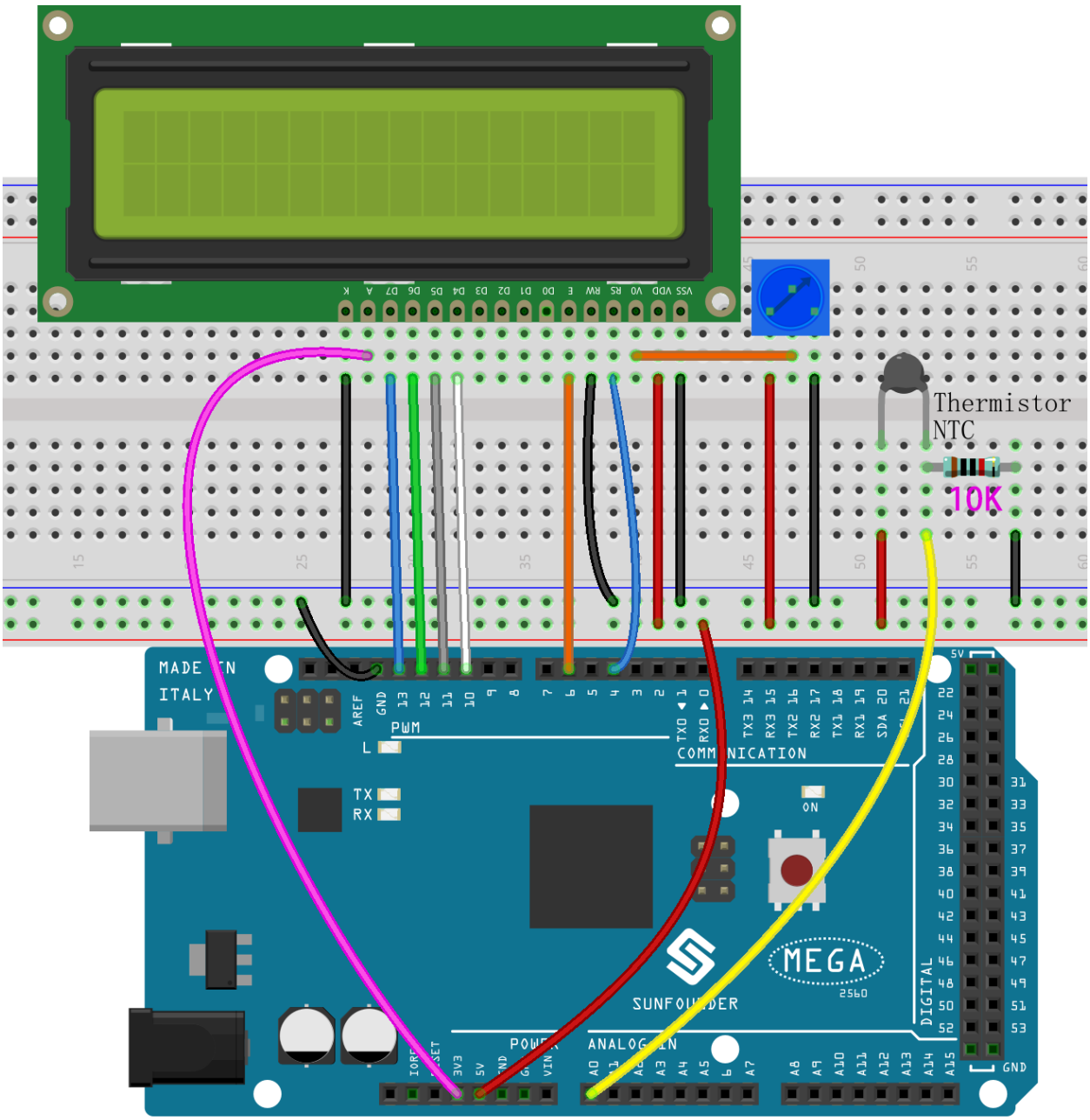
然后用表达式替换关系式 R = (5 x 10k /V) - 10k 中的 V，我们可以得到：R=1025 x 10k/a-10k。

最后将这里的公式中的 R 代入 T= B/[ln(R/ 10) + (B/ 298)] –273，就是刚刚形成的。然后我们最终得到温度的关系如下：

T = B/[ln{[1025 X 10/a) - 10]/10} 十 (B/298)] –273

7.12.4 实验步骤

第 1 步：搭建电路。

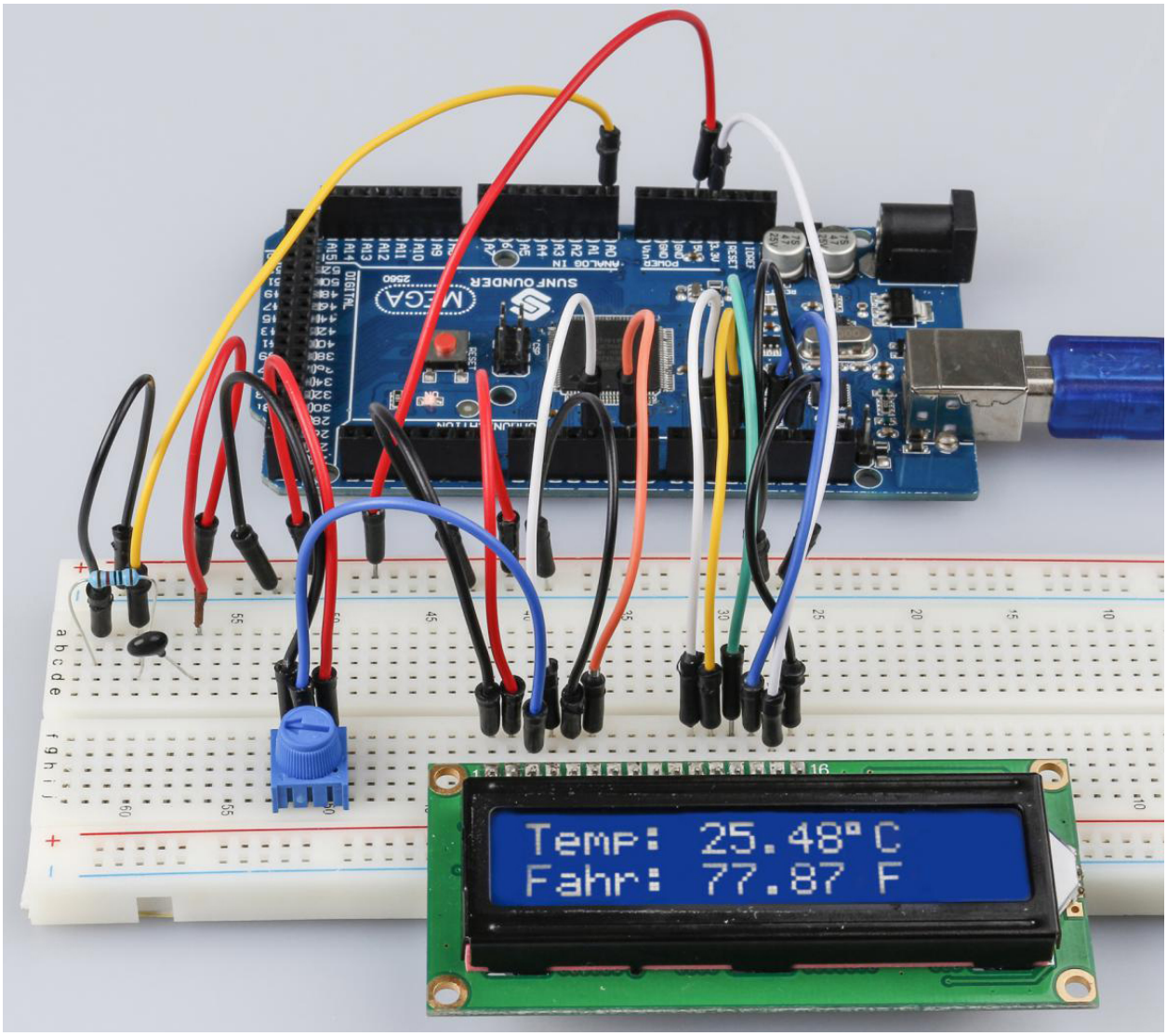


第 2 步：打开代码文件 `Lesson_12_Thermistor.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在你可以在 LCD1602 上显示在摄氏度和华氏度下的温度。



7.12.5 代码

7.12.6 代码分析

设置变量

```
#define analogPin A0 //the thermistor attach to
#define beta 3950 //the beta of the thermistor
#define resistance 10 //the value of the pull-up resistor
```

设置 β 系数的值，在热敏电阻的数据表中有描述。

获取温度

```
void loop()
{
    //read thermistor value
    long a = analogRead(analogPin);
```

(续下页)

(接上页)

```
    //the calculating formula of temperature
    float tempC = beta / (log((1025.0 * 10 / a - 10) / 10) + beta / 298.0) - 273.0;
    float tempF = 1.8 * tempC + 32.0;
```

读取 A0 的值（热敏电阻），然后通过公式计算出摄氏温度，再通过公式将摄氏温度转换为华氏温度。

在 LCD1602 上显示温度

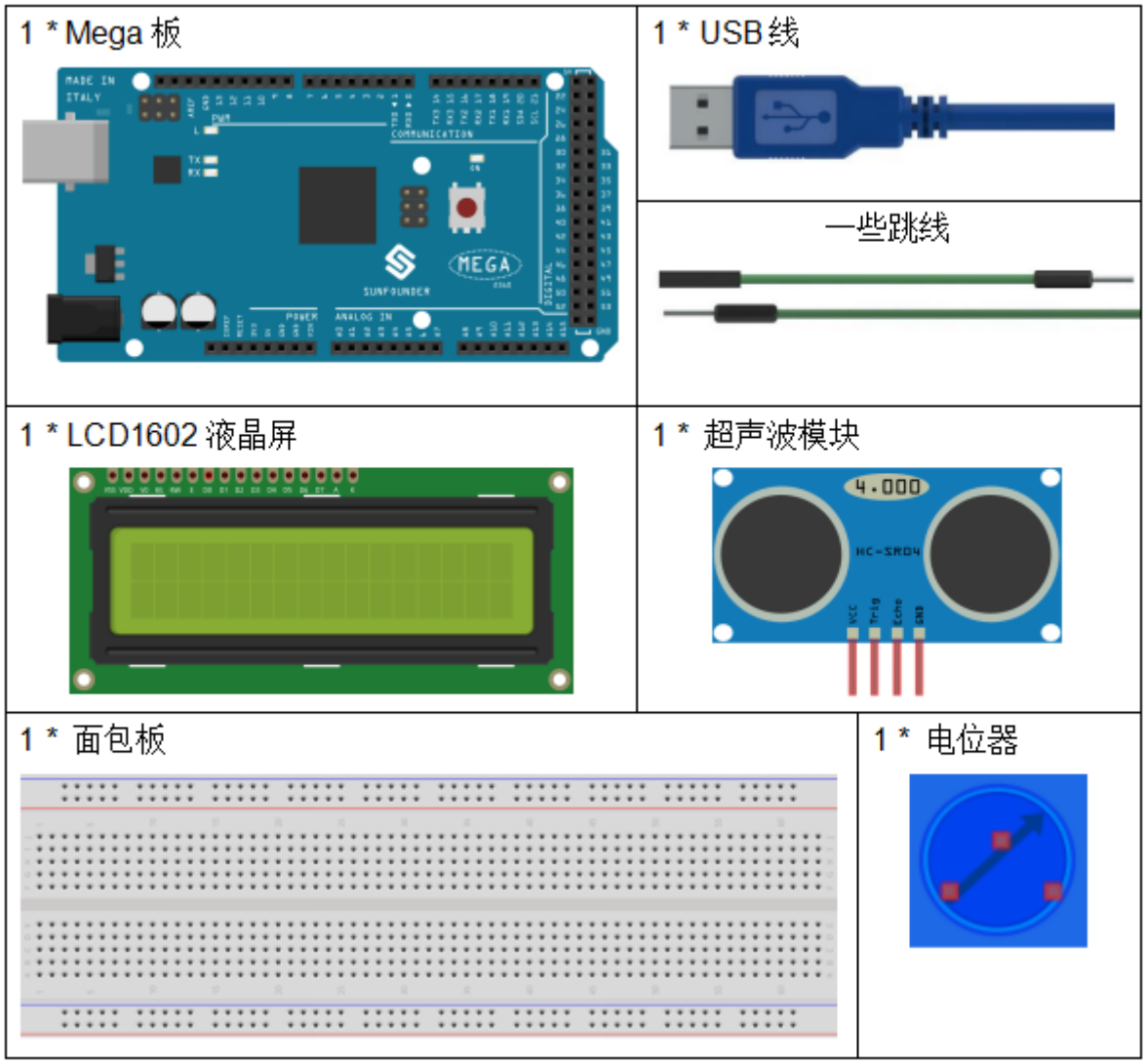
```
    lcd.setCursor(0, 0); // set the cursor to column 0, line 0
    lcd.print("Temp: ");// Print a message of "Temp: "to the LCD.
    // Print a centigrade temperature to the LCD.
    lcd.print(tempC);
    // Print the unit of the centigrade temperature to the LCD.
    lcd.print(char(223));//print the unit" ℃ "
    lcd.print("C");
    // (note: line 1 is the second row, since counting begins with 0):
    lcd.setCursor(0, 1); // set the cursor to column 0, line 1
    lcd.print("Fahr: ");
    lcd.print(tempF);// Print a Fahrenheit temperature to the LCD.
    lcd.print(" F"); // Print the unit of the Fahrenheit temperature to the LCD.
    delay(200); //wait for 100 milliseconds
}
```

7.13 第 13 课超声波

7.13.1 介绍

倒车时，你会看到汽车与周围障碍物之间的距离，以避免碰撞。检测距离的装置是超声波传感器。在本实验中，你将了解超声波如何检测距离。

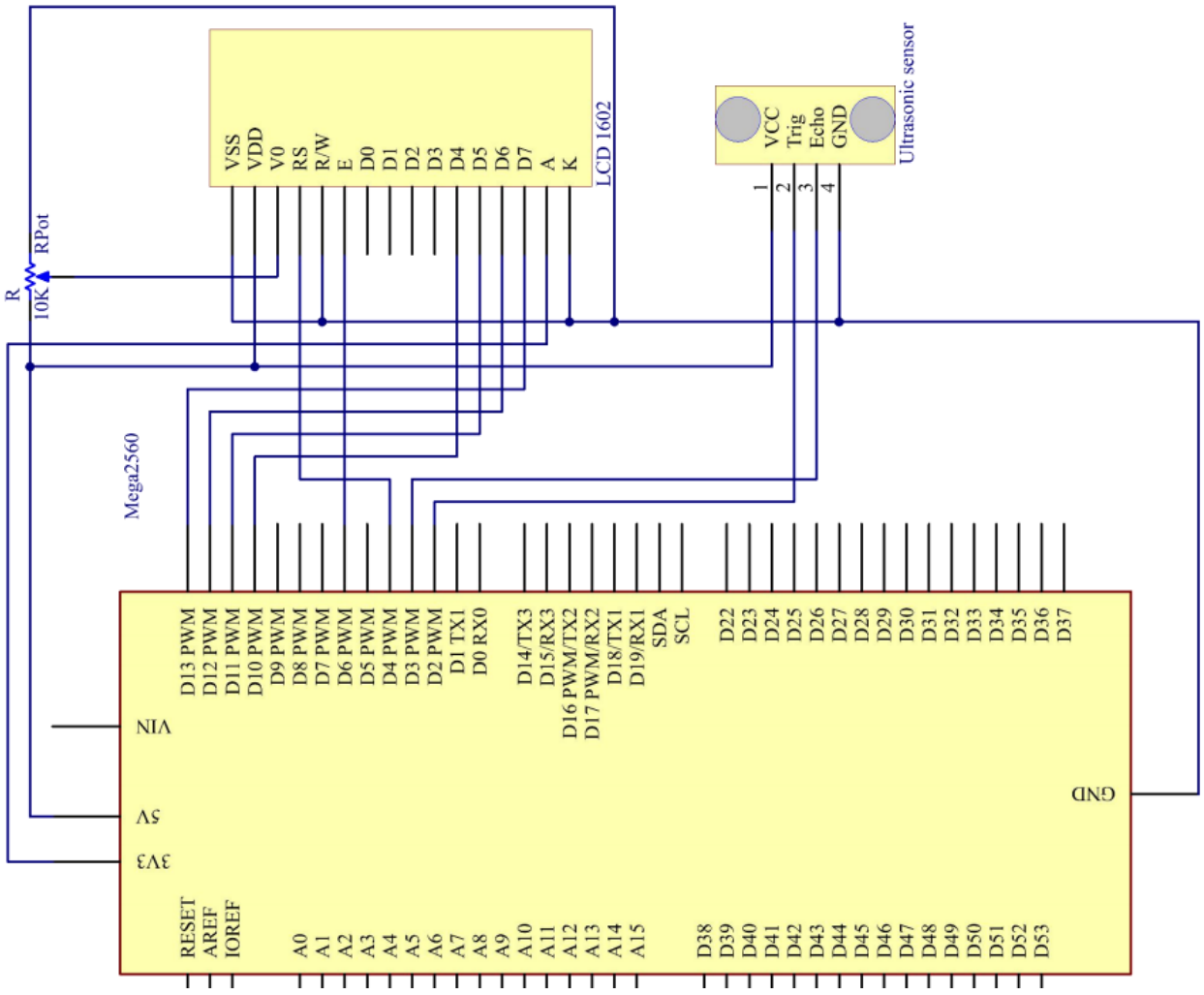
7.13.2 所需器件



- *SunFounder Mega 板*
- 面包板
- 跳线
- 电位器
- 超声波模块
- *LCD1602 液晶显示屏*

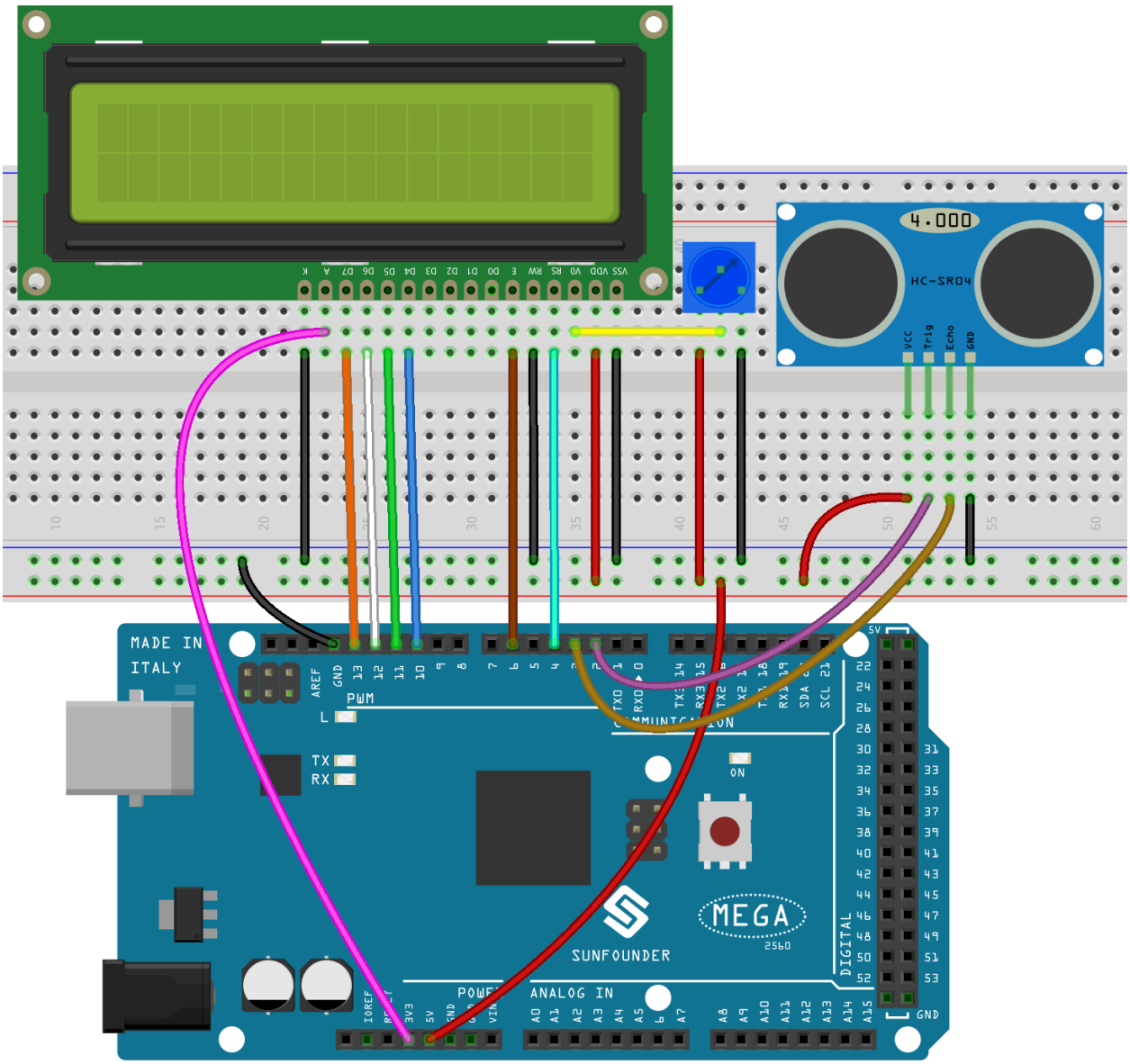
7.13.3 原理图

原理图如下所示：



7.13.4 实验步骤

第 1 步：搭建电路。



第 2 步：打开代码文件 `Lesson_13_Ultrasonic.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

备注： 如果出现如下错误，是因为你没有添加名为 NewPing 的库，请参考添加库。

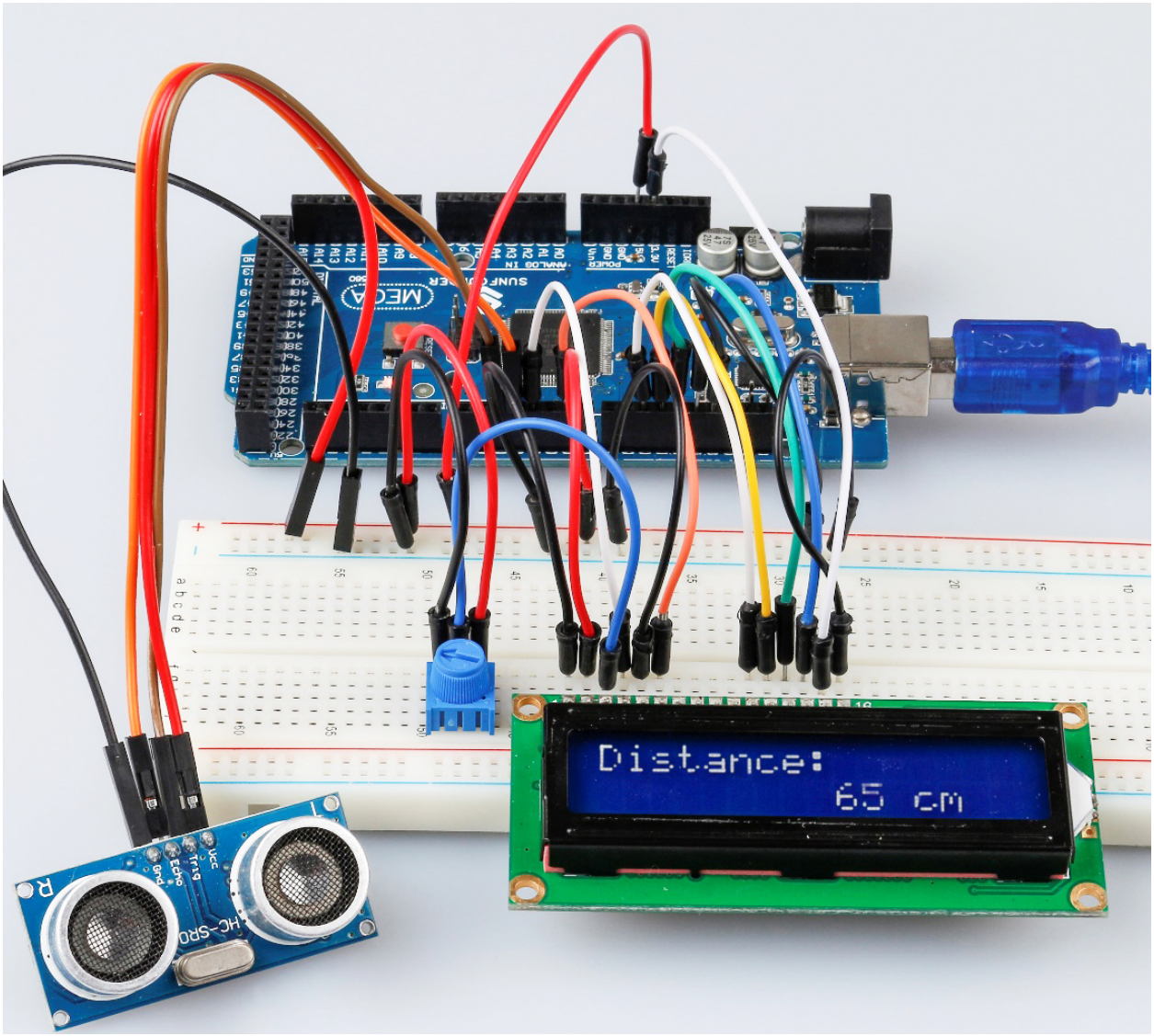
```
12  #include <NewPing.h>
```

```
NewPing.h: No such file or directory                    Copy error messages
"D:\\Program Files (x86)\\Arduino\\hardware\\tools\\avr/bin/avr-g++" -
Ultrasonic:12:21: error: NewPing.h: No such file or directory

compilation terminated.

Using library LiquidCrystal at version 1.0.7 in folder: D:\Program Fil
```

现在，如果你使用一张纸靠近或远离传感器。你会看到 LCD1602 上显示的值发生变化，这表示纸张与超声波传感器之间的距离。



7.13.5 代码

7.13.6 代码分析

初始化超声波和 LCD1602

```
#include <LiquidCrystal.h>
#include <NewPing.h>

LiquidCrystal lcd(4, 6, 10, 11, 12, 13);//lcd(RS,E,D4,D5,D6,D7)

#define TRIGGER_PIN  2  //  trig pin on the ultrasonic sensor attach to pin2 .
#define ECHO_PIN     3  //  echo pin on the ultrasonic sensor attach to pin3.
#define MAX_DISTANCE 400 // Maximum distance we want to ping for (in centimeters).␣
↪Maximum sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and␣
↪maximum distance.
```

创建一个 NewPing 变量 sonar。NewPing 的基本格式为：NewPing(uint8_t trigger_pin, uint8_t echo_pin, int max_cm_distance) 。这里 uint 表示无符号整数，8 表示 8 位。所以这里 uint8 格式的值意味着一个 unsigned-char 类型的值。

将时间转换成距离

```
unsigned int uS = sonar.ping(); // Send ping, get ping time in
microseconds (uS).
```

ping() 用来计算从脉冲发送到接收的时间。定义一个变量 Us 来存储接收的时间，单位应该是微秒（us）。

```
int distance = uS / US_ROUNDTRIP_CM;
```

uS / US_ROUNDTRIP_CM `` 是将 ``ping() 发送和接收之间的时间转换为距离的公式，单位是厘米。

在 LCD1602 上显示距离

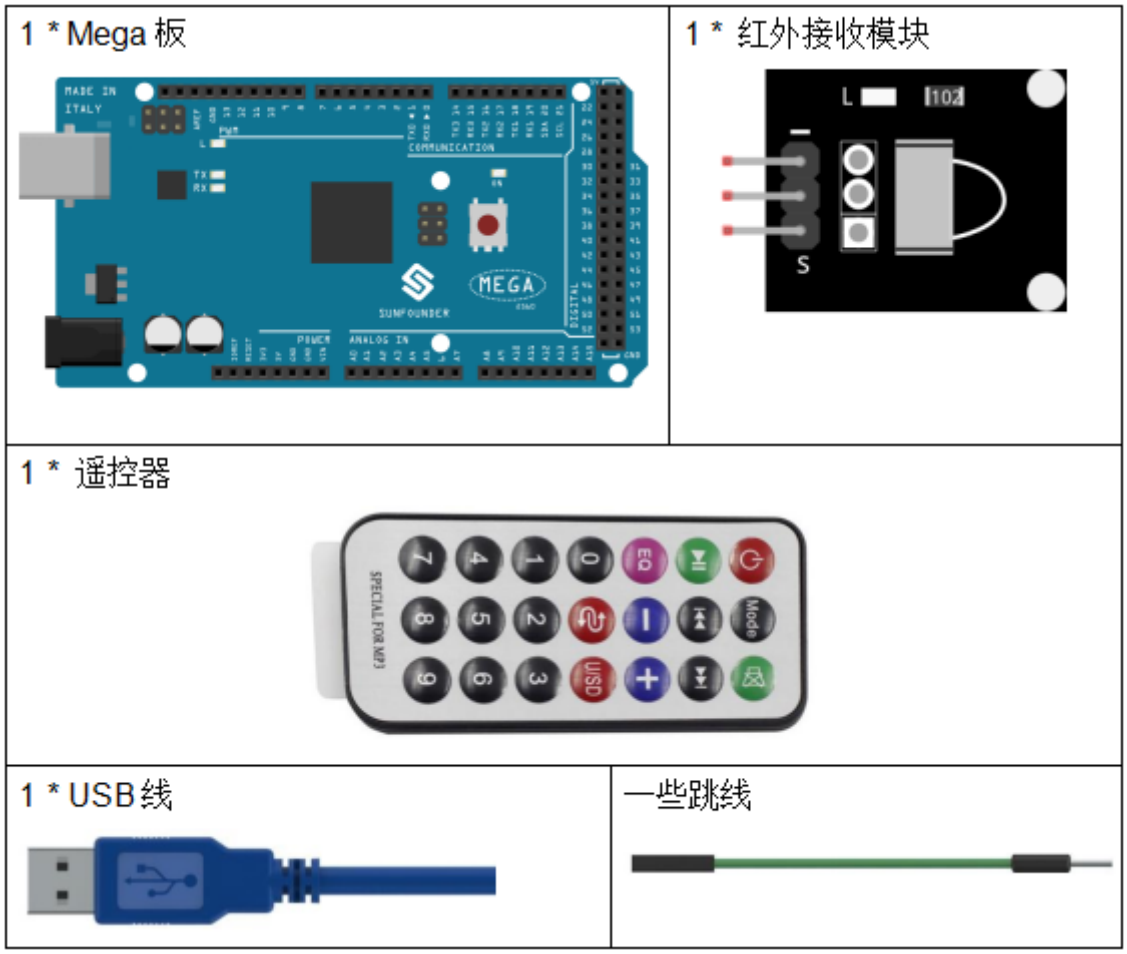
```
lcd.setCursor(0, 0);//Place the cursor at Line 1, Column 1. From here the characters␣
↪are to be displayed
lcd.print("Distance:");////Print Distance: on the LCD
lcd.setCursor(0, 1);//Set the cursor at Line 1, Column 0
lcd.print("             ");//Here is to leave some spaces after the characters so as␣
↪to clear the previous characters that may still remain.
lcd.setCursor(9, 1);//Set the cursor at Line 1, Column 9.
lcd.print(distance);// print on the LCD the value of the distance converted from the␣
↪time between ping sending and receiving.
lcd.setCursor(12, 1);//Set the cursor at Line 1, Column 12.
lcd.print("cm");//print the unit "cm"
```

7.14 第 14 课红外接收模块

7.14.1 介绍

红外接收器是接收红外信号并能独立接收红外线并输出兼容 TTL 电平的信号的部件。它的尺寸与普通的塑料封装晶体管相似，适用于各种红外遥控和红外传输。

7.14.2 所需器件

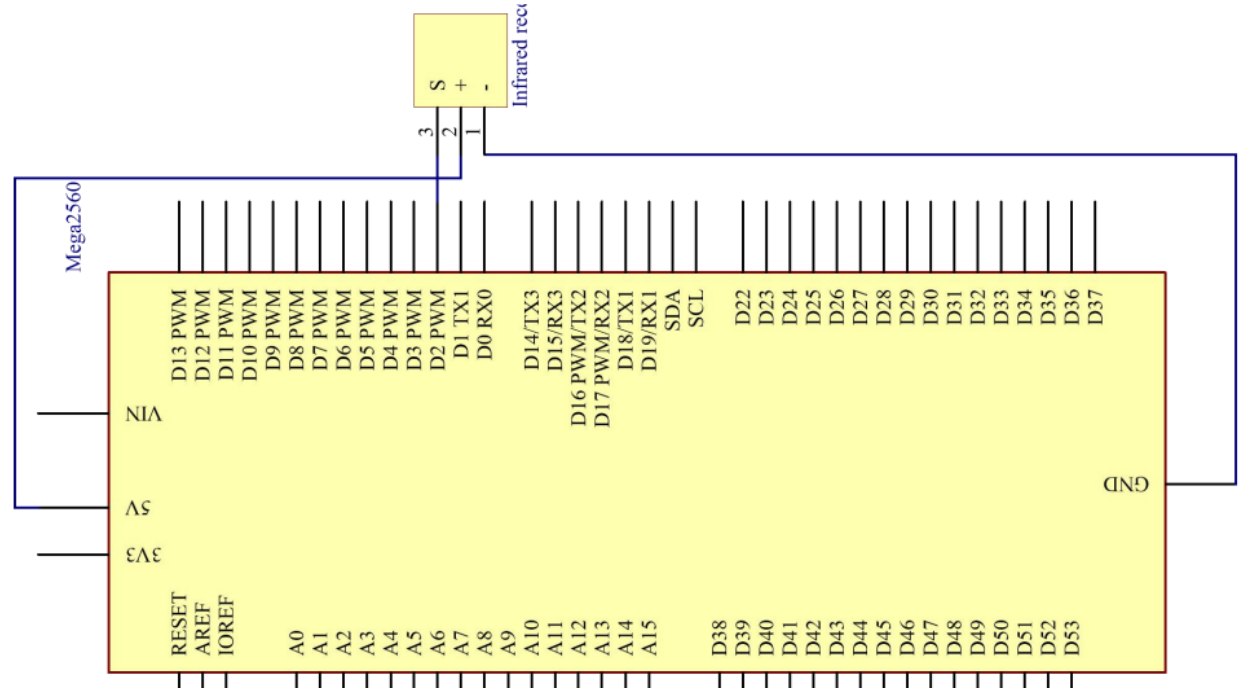


- *SunFounder Mega 板*
- 面包板
- 跳线
- 红外接收模块

7.14.3 原理图

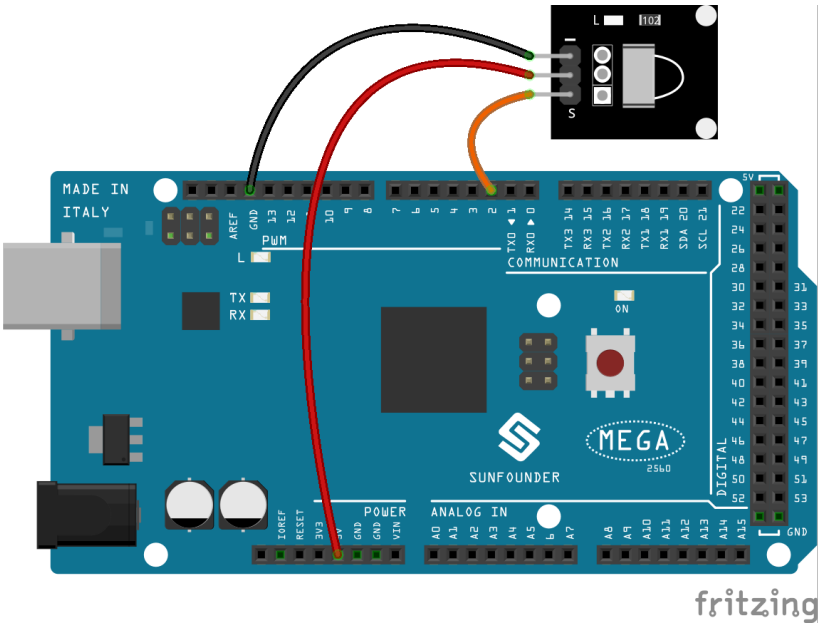
通过编程读取遥控上的某个键的键值（例如，电源键）。当你按下该键时，红外线会从遥控器发出并被红外线接收器接收，控制板上的 LED 会亮起。

原理图如下所示：



7.14.4 实验步骤

第 1 步：搭建电路。



第 2 步：打开代码文件 `Lesson_14_Infrared_Receiver.ino`。

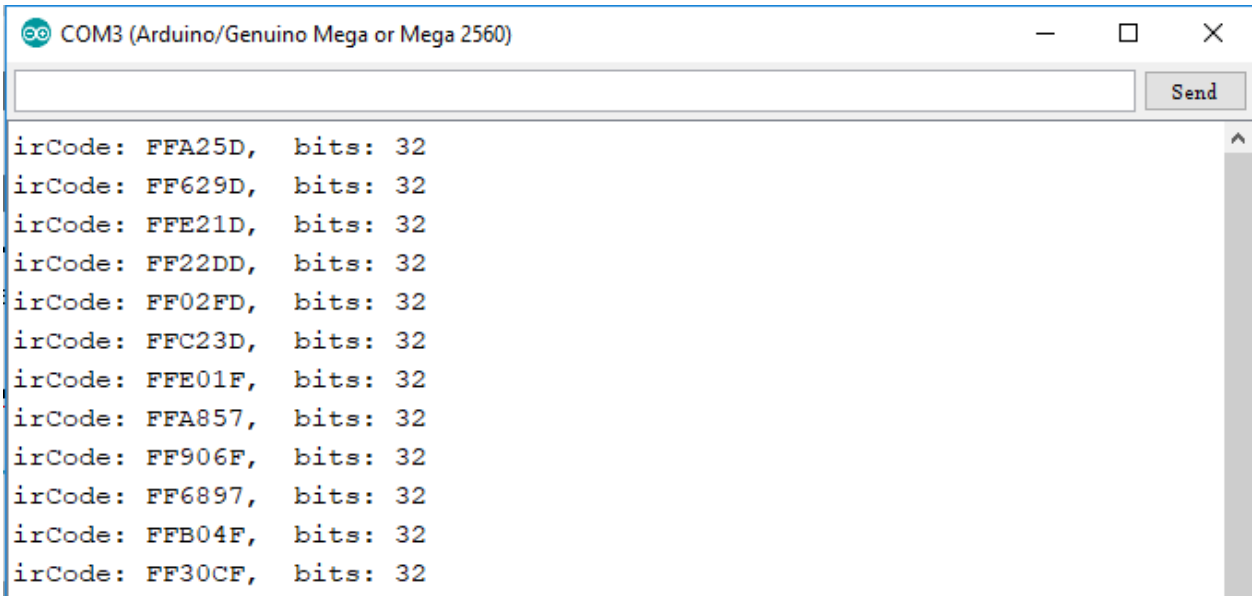
第 3 步：选择 **开发板**和 **端口**。

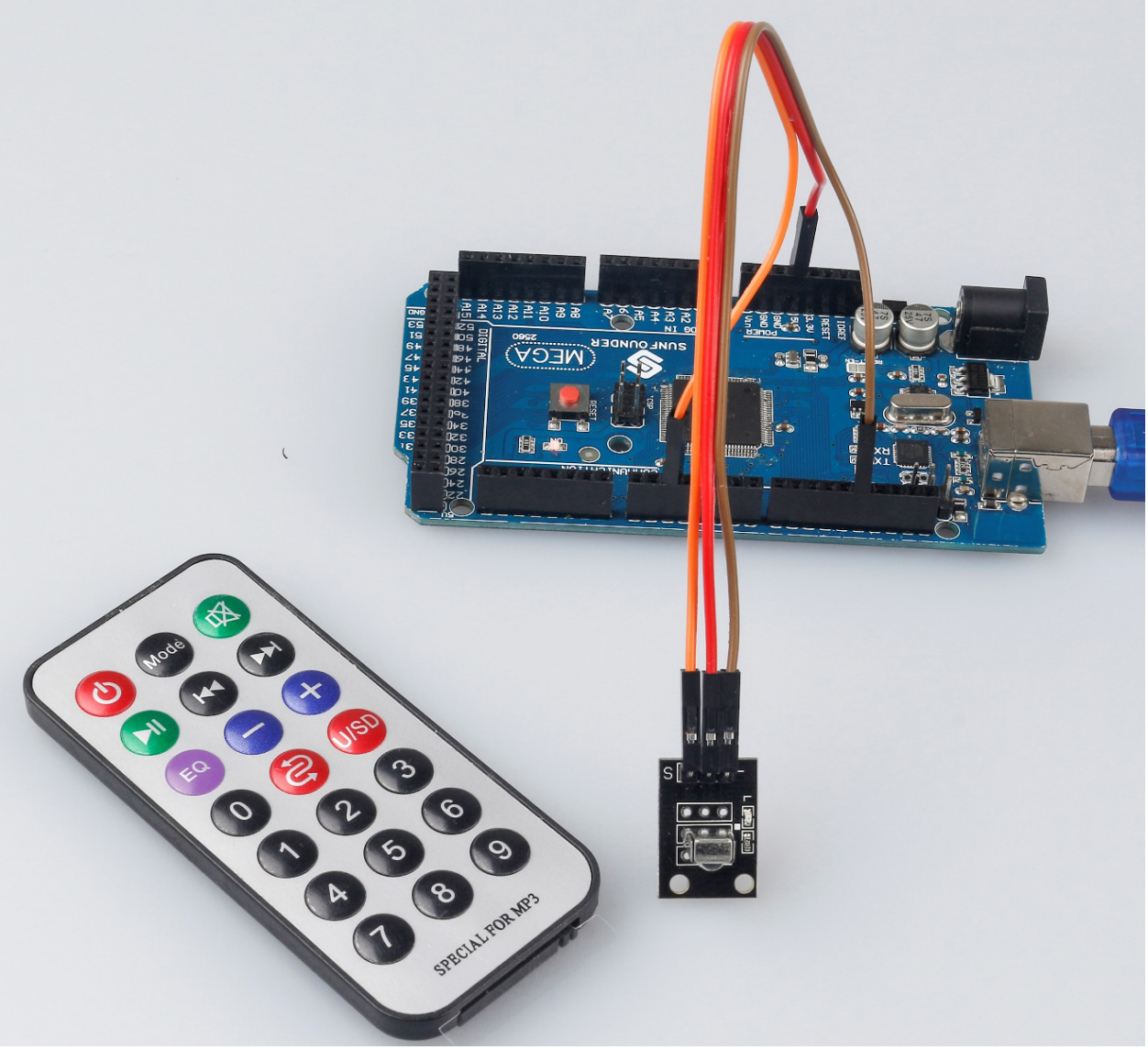
第 4 步：点击 **上传**按钮来上传代码。

现在，按下遥控器上的电源，控制板上连接到引脚 13 的 LED 将亮起。如果按其他键，LED 将熄灭。

备注:

- 遥控器的尾部有一块透明塑料片用来切断电源，你需要在使用前拔出。
- 请轻轻按下遥控器上的按钮，以避免无效数据 FFFFFFFF。





7.14.5 代码

7.14.6 代码分析

初始化红外接收器

```
#include <IRremote.h>
const int irReceiverPin = 2; // the infrared-receiver attact to pin2
const int ledPin = 13; // built-in LED
IRrecv irrecv(irReceiverPin); // Initialize the infrared-receiver
decode_results results; // The decoding result is placed in the result of the decode␣
↪results structure.
```

启用红外接收器

```
irrecv.enableIRIn(); // Restart the receiver
```

接收并打印数据

```
if (irrecv.decode(&results)) { // If receive a data
```

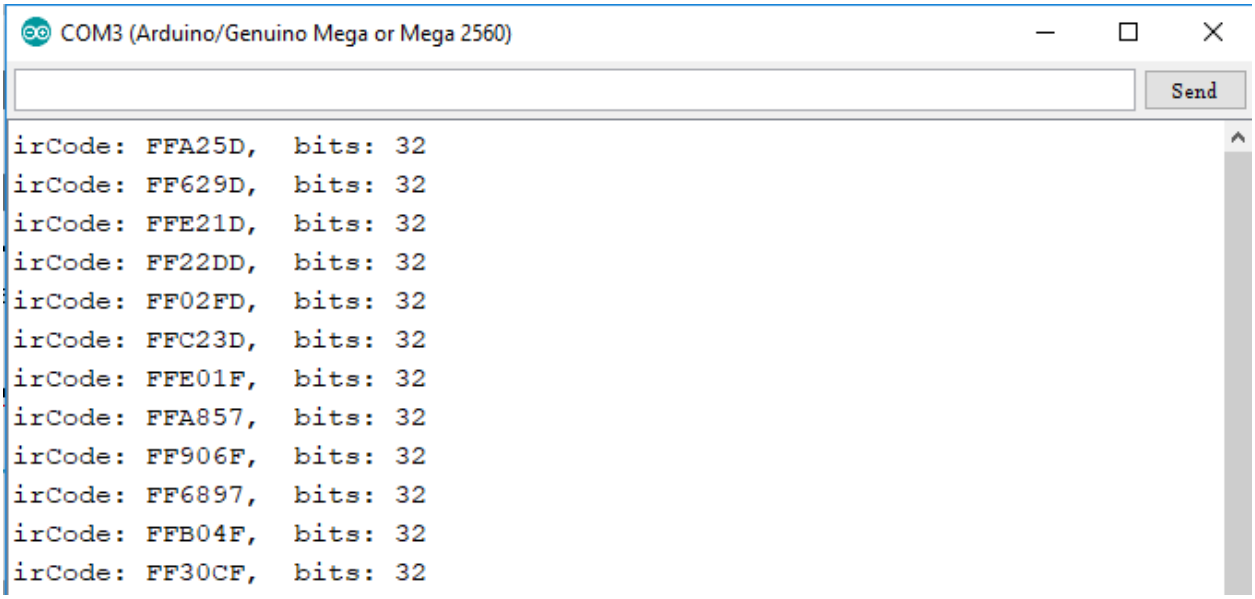
decode(&results)：对接收到的红外信息进行解码，没有数据返回 0，否则返回 1。解码结果存放在 results 中。

```
    Serial.print("irCode: "); // print "irCode: " on the serial monitor
    Serial.print(results.value, HEX); // print the signal on serial monitor
    in hexadecimal
    Serial.print(", bits: ");
    Serial.println(results.bits); // Print the data bits
    irrecv.resume(); // Receive next data
}
delay(600);
```

如果电源键被按下

```
if(results.value == 0xFFA25D) // if the power button on the remote control is pressed
```

0xFFA25D 是遥控器电源键的代码，如果你想定义其他按钮，你可以从串口监视器上读取每个按键的代码。




```
{
    digitalWrite(ledPin,HIGH); // Turn on the LED
}
else
{
    digitalWrite(ledPin,LOW); // else turn of the LED
}
```

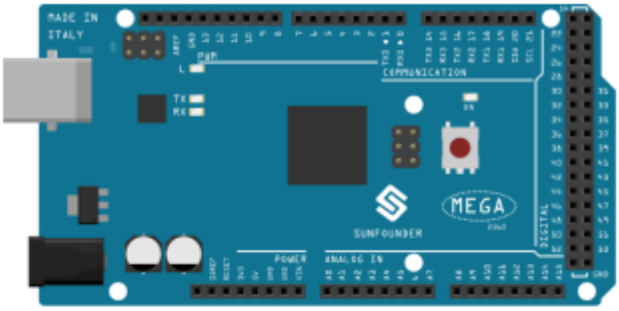
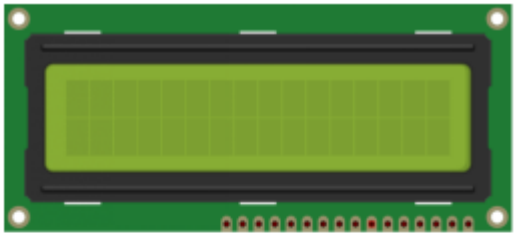
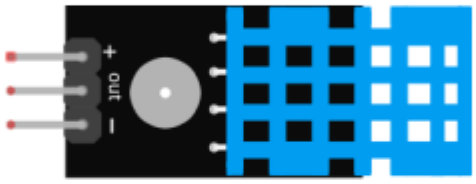
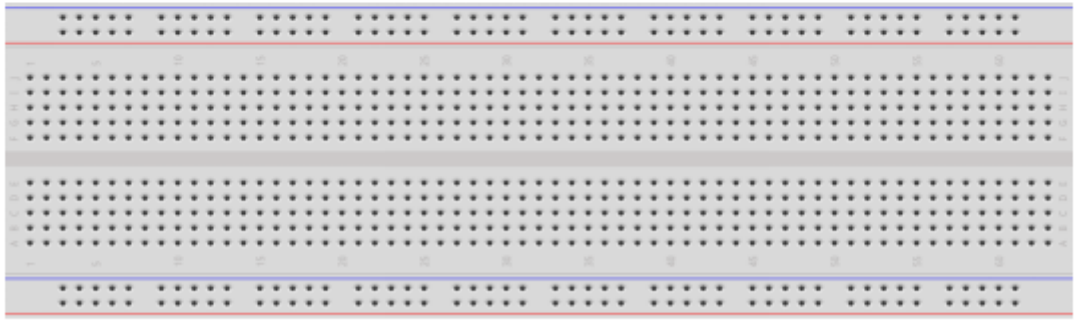
7.15 第 15 课温湿度传感器

7.15.1 介绍

数字温湿度传感器 DHT11 是一种复合传感器，包含经过校准的温湿度数字信号输出。采用专用数字模块采集技术和温湿度传感技术，确保产品具有高可靠性和优异的长期稳定性。

该传感器包括电阻感湿元件和 NTC 测温装置，并与高性能 8 位微控制器相连。

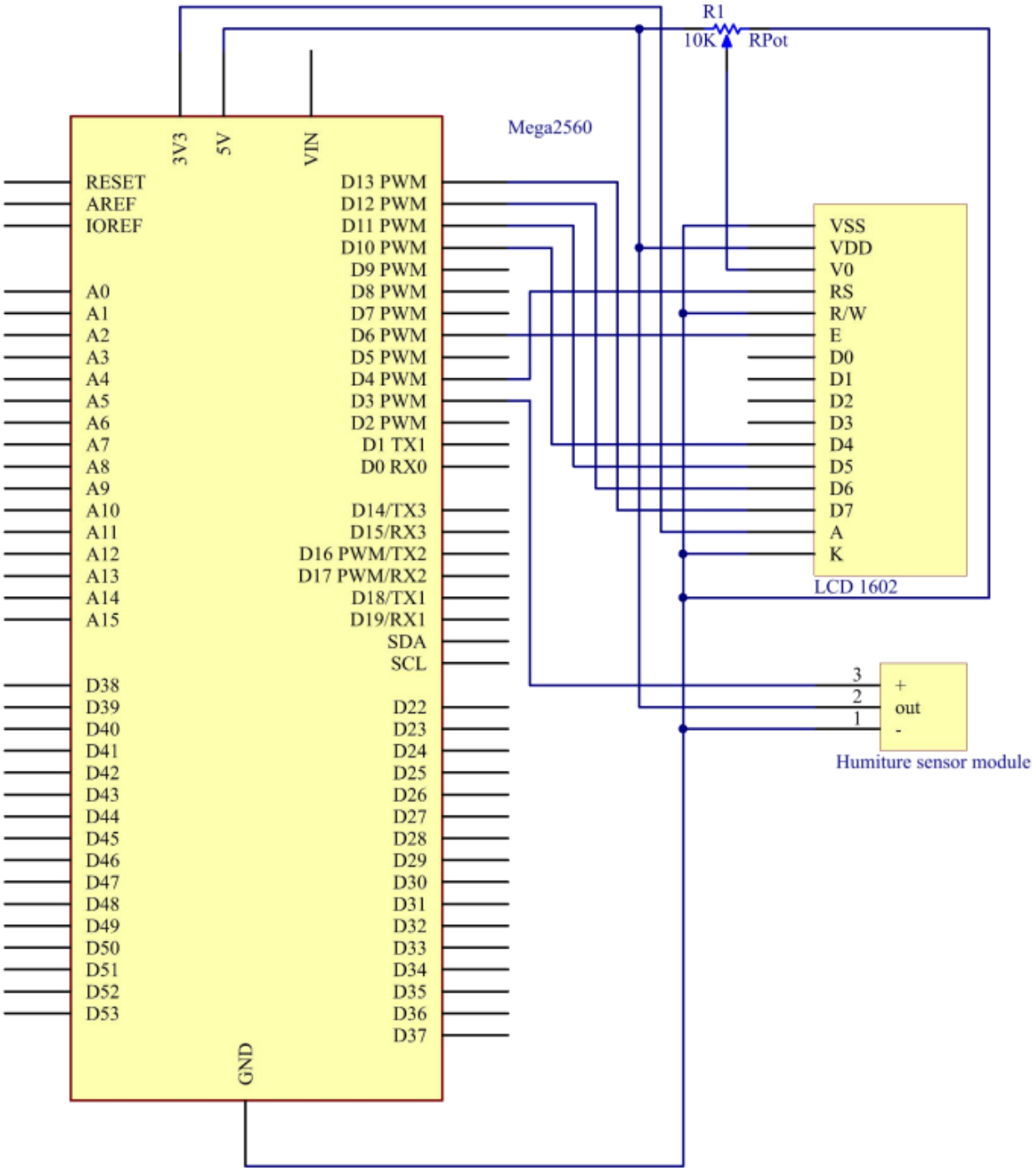
7.15.2 所需器件

<table>
<tr><td>1 * Mega 板
  </td><td>1 * LCD1602 液晶屏
  </td></tr>
<tr><td>1 * 电位器
  </td><td>1 * 温湿度传感器模块
  </td></tr>
<tr><td colspan="2">1 * 面包板
  </td></tr>
<tr><td>1 * USB 线
  </td><td>一些跳线
  </td></tr>
</table>

- *SunFounder Mega* 板
- 面包板
- 跳线
- *LCD1602* 液晶显示屏
- 电位器
- 温湿度传感器模块

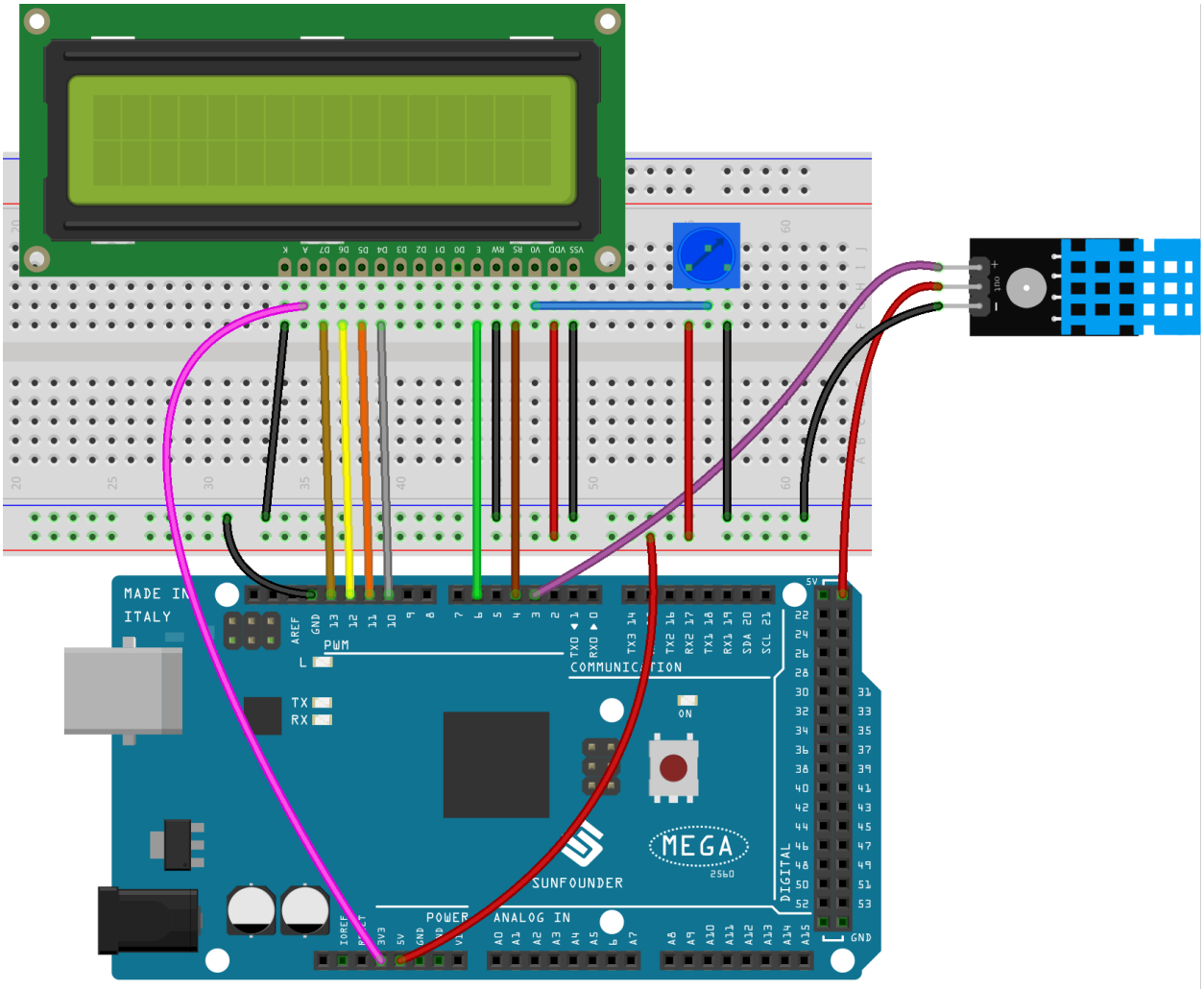
7.15.3 原理图

原理图如下所示：


R1
10K
RPot
Mega2560
3V3
5V
VIN
RESET
AREF
IOREF
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
D38
D39
D40
D41
D42
D43
D44
D45
D46
D47
D48
D49
D50
D51
D52
D53
GND
D13 PWM
D12 PWM
D11 PWM
D10 PWM
D9 PWM
D8 PWM
D7 PWM
D6 PWM
D5 PWM
D4 PWM
D3 PWM
D2 PWM
D1 TX1
D0 RX0
D14/TX3
D15/RX3
D16 PWM/TX2
D17 PWM/RX2
D18/TX1
D19/RX1
SDA
SCL
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
D32
D33
D34
D35
D36
D37
VSS
VDD
V0
RS
R/W
E
D0
D1
D2
D3
D4
D5
D6
D7
A
K
LCD 1602
3
2
1
+
out
-
Humiture sensor module

7.15.4 实验步骤

第 1 步：搭建电路。

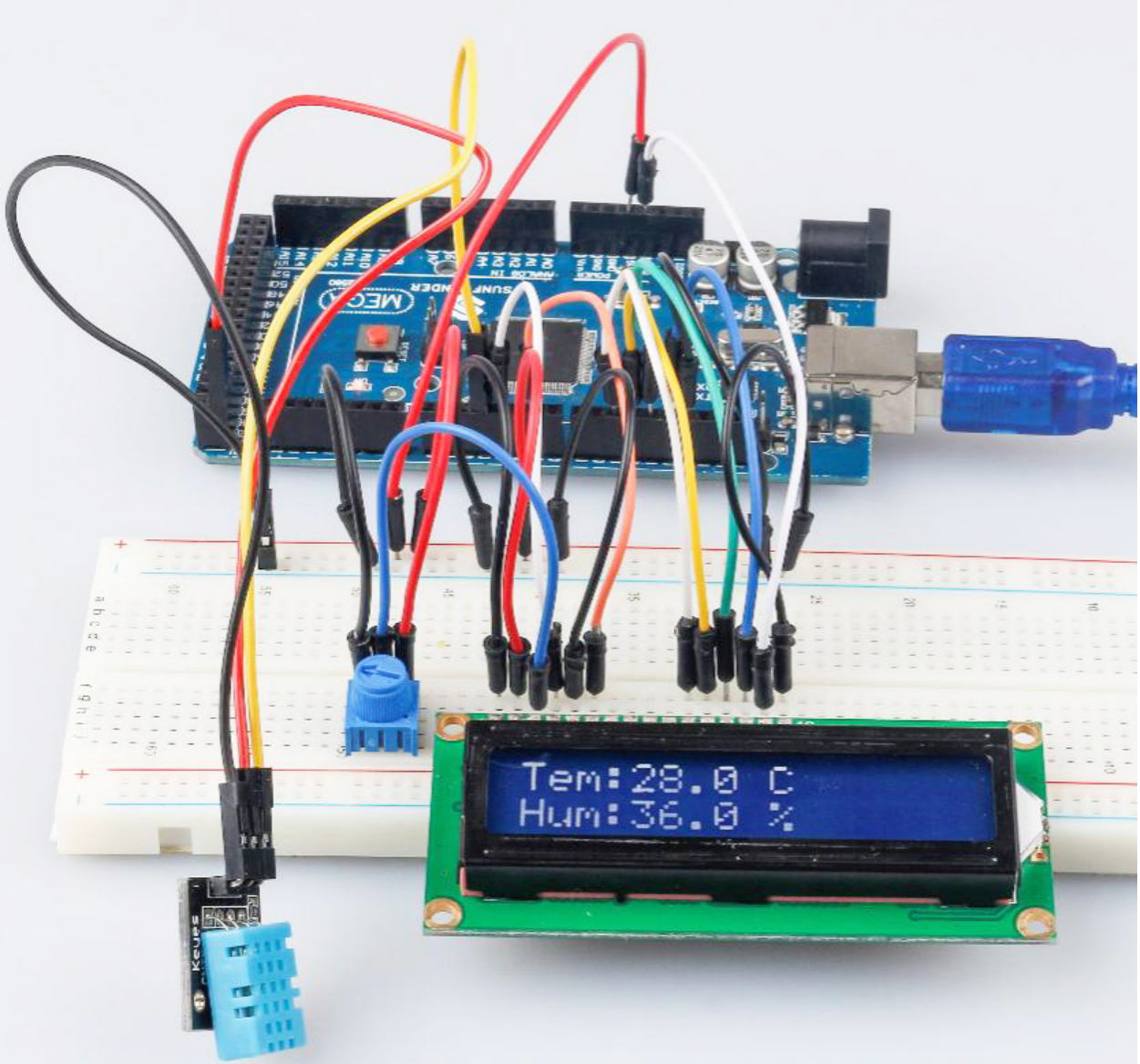


第 2 步：打开代码文件 `Lesson_15_Humiture_Sensor.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你可以看到 LCD1602 上显示的当前湿度和温度值。



7.15.5 代码

7.15.6 代码分析

初始化温湿度和 LCD1602

```
#include <dht.h> // Include the head file dht.h
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 6, 10, 11, 12, 13); // initialize the LCD1602
dht DHT;
#define DHT11_PIN 3 // the humiture sensor attact to pin3
```

读温湿度传感器的值

```
int chk = DHT.read11(DHT11_PIN);
switch (chk)
{
    case DHTLIB_OK:
        Serial.println("OK,\t");
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.println("Checksum error,\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.println("Time out error,\t");
        break;
    default:
        Serial.println("Unknown error,\t");
        break;
}
```

使用该 `read11()` 函数读取温湿度传感器的值。如果串口监视器上显示 OK，则说明温湿度传感器工作正常。

- `read11()`：返回值：

```
// DHTLIB_OK: Indicate the humiture sensor is work well.
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
```

LCD1602 上的显示

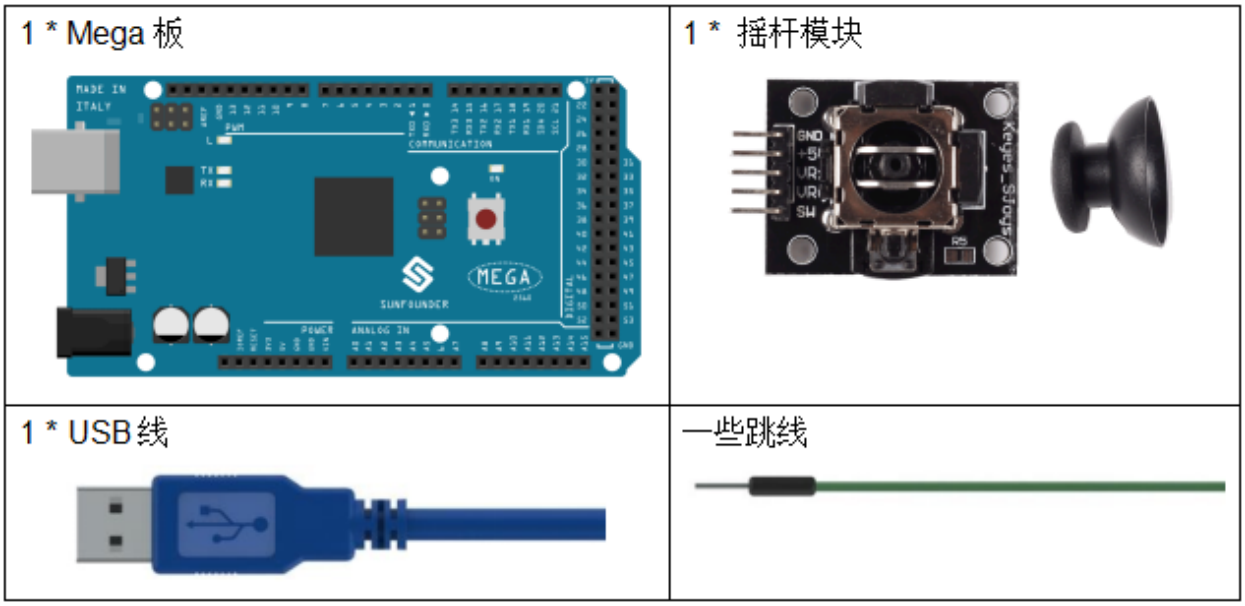
```
lcd.setCursor(0, 0);
lcd.print("Tem:");
lcd.print(DHT.temperature,1); //print the temperature on lcd
lcd.print(" C");
lcd.setCursor(0, 1);
lcd.print("Hum:");
lcd.print(DHT.humidity,1); //print the humidity on lcd
lcd.print(" %");
delay(200); //wait a while
```

7.16 第 16 课摇杆

7.16.1 介绍

操纵杆是一种输入设备，由一个在底座上旋转的操纵杆组成，并向它所控制的设备报告其角度或方向。操纵杆通常用于控制视频游戏和机器人，此处使用操纵杆 PS2。

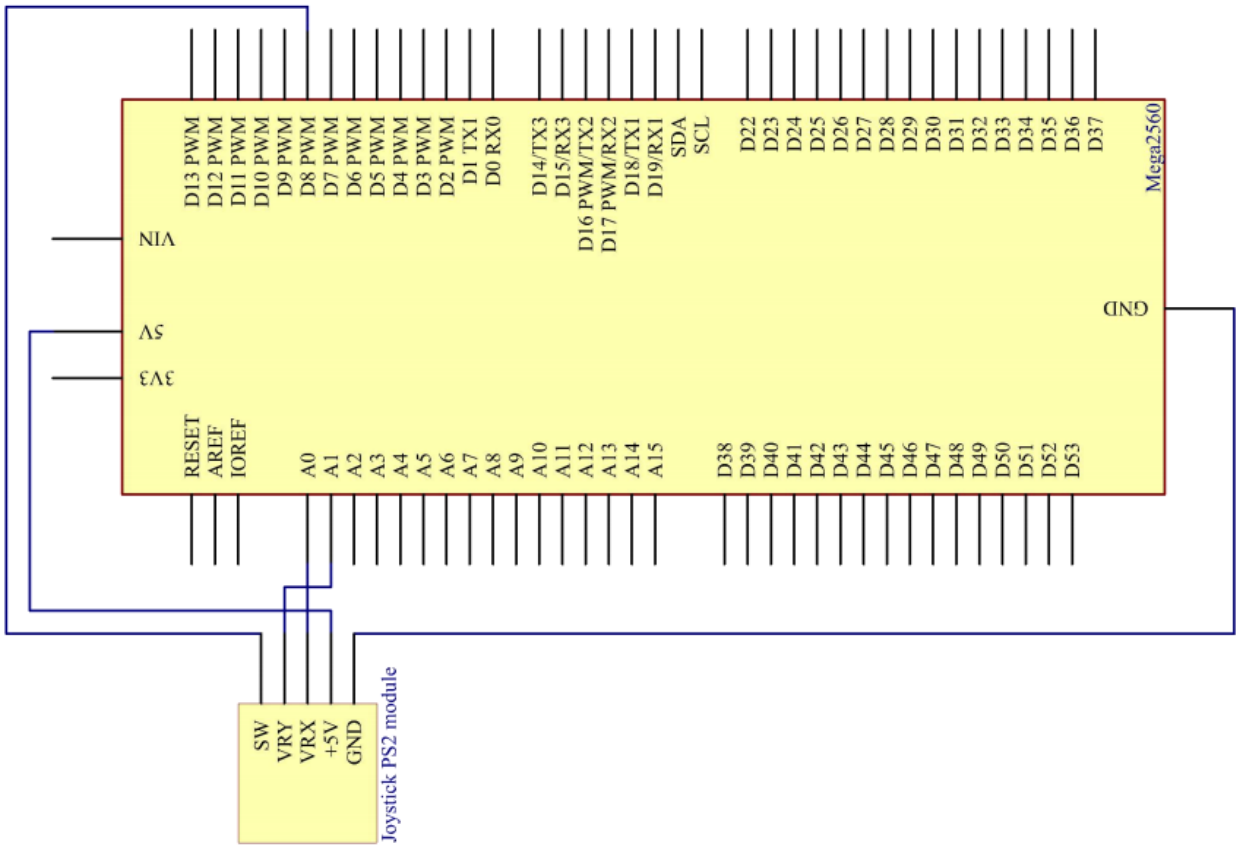
7.16.2 所需器件



- *SunFounder Mega 板*
- 面包板
- 跳线
- 摇杆模块

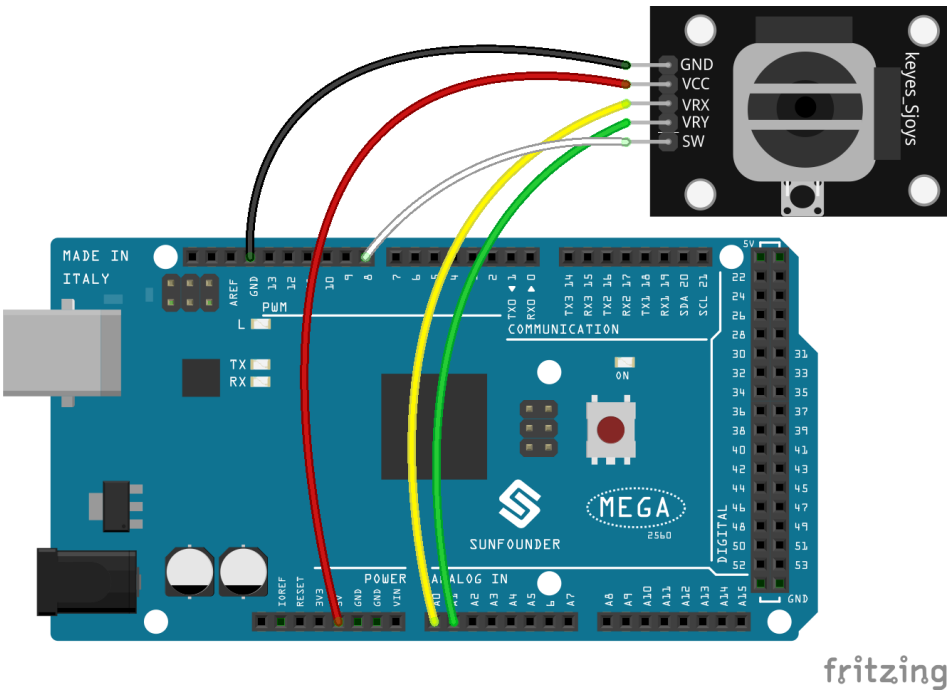
7.16.3 原理图

该模块有 2 路模拟量输出（对应 X，Y 双轴偏移）和 1 路数字输出（Z 轴），用来表示摇杆是否被按压。
在这个实验中，将在串口监视器中显示摇杆上的 X,Y 和 Z 轴上的值的变化。
原理图如下所示：



7.16.4 实验步骤

第 1 步：搭建电路。

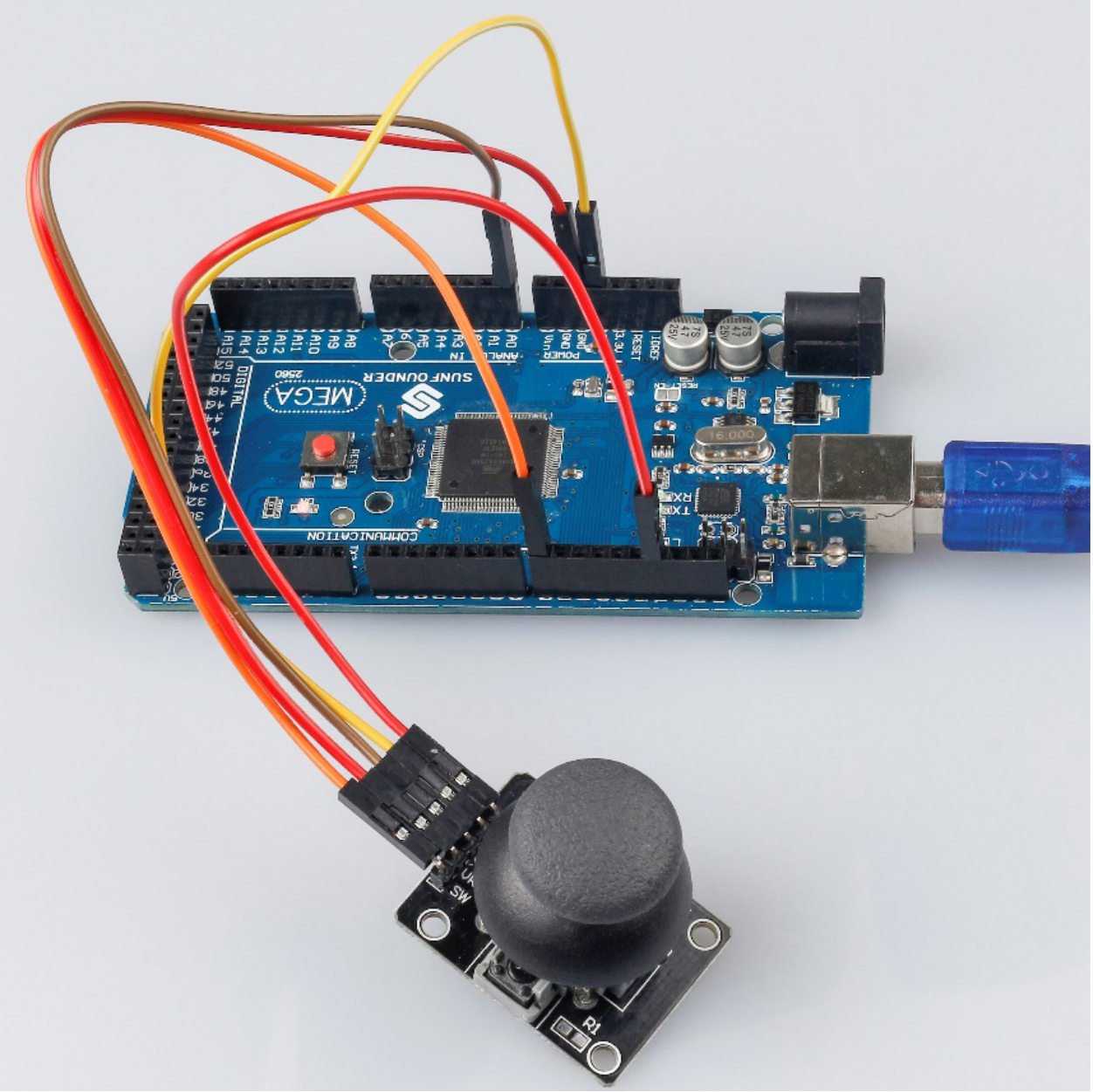


第 2 步：打开代码文件 `Lesson_16_Joystick_PS2.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，拨动摇杆，串口监视器上显示的 X 和 Y 轴坐标会相应改变；按摇杆，会显示 Z=0。



7.16.5 代码

7.16.6 代码分析

该代码使用串行监视器打印操纵杆 ps2 的 VRX、VRY 和 SW 引脚的值。

```
void loop()
{
    Serial.print("X: ");
    Serial.print(analogRead(xPin), DEC);  // print the value of VRX in DEC
    Serial.print("|Y: ");
    Serial.print(analogRead(yPin), DEC);  // print the value of VRX in DEC
    Serial.print("|Z: ");
    Serial.println(digitalRead(swPin));   // print the value of SW
    delay(500);
}
```

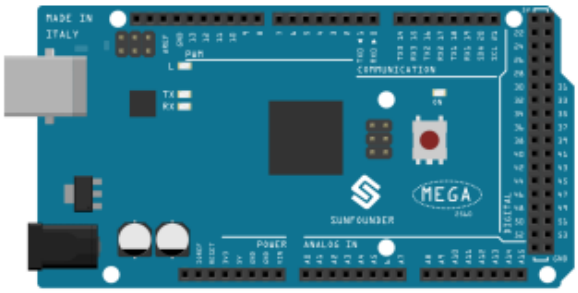
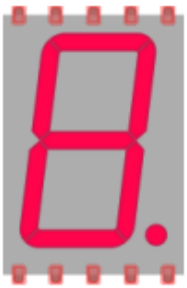
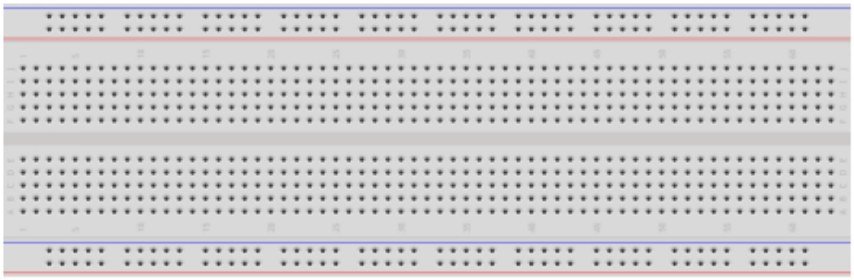
- `"|Y: "` 中的 `|` 用来隔开数据。

7.17 第 17 课 7 段数码管

7.17.1 介绍

7 段数码管是一种可以显示数字和字母的设备。它由七个并联的 LED 组成。通过将数码管上的引脚连接到电源并启用相关引脚，从而打开相应的 LED 段，可以显示不同的字母/数字。在本课中，我们将学习如何在其上显示特定字符 。

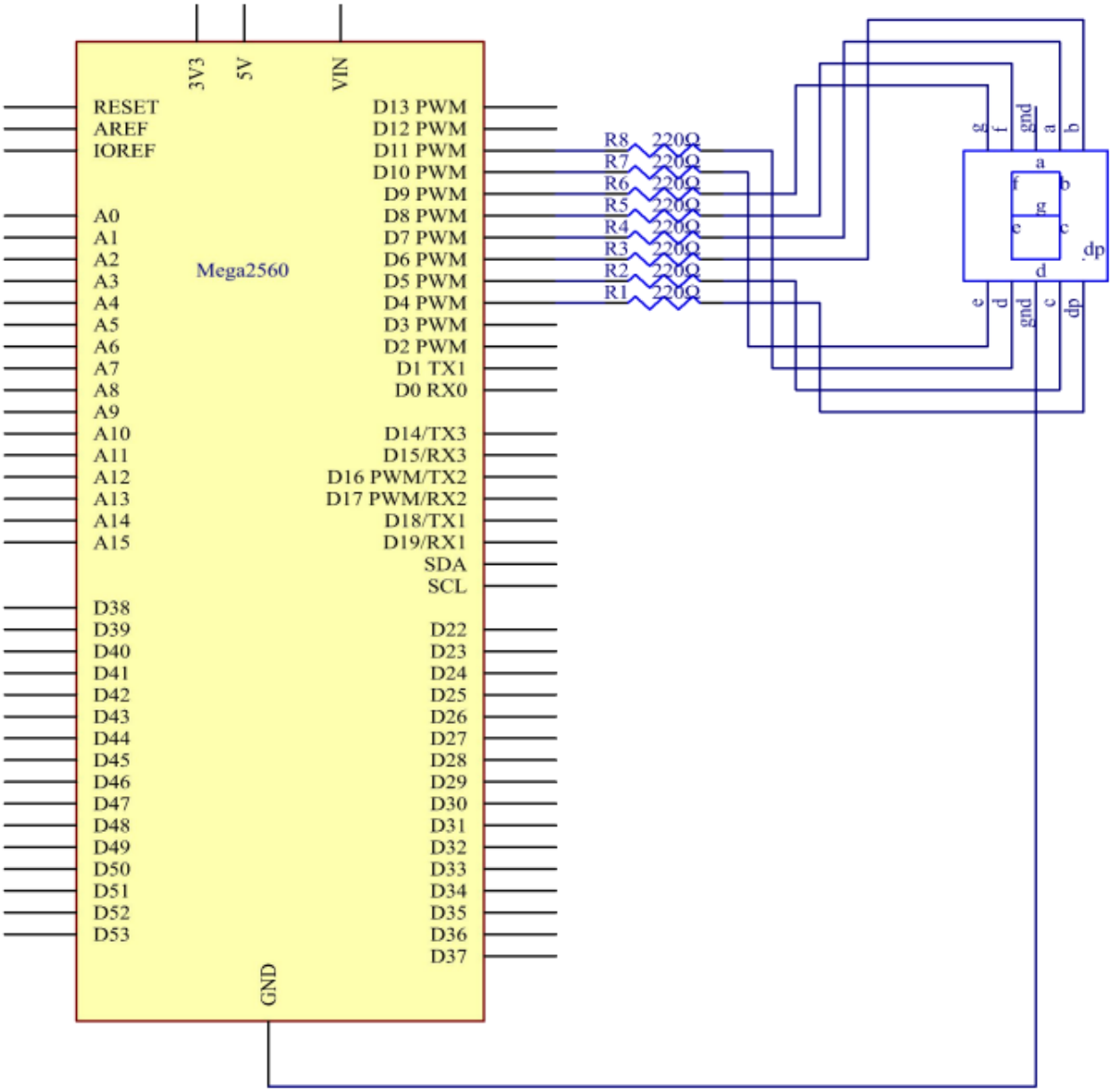
7.17.2 所需器件

<table>
<tr><td>1 * Mega 板
  </td><td>8 * 电阻 (220Ω)
  </td><td>1 * 7 段数码管
Common Cathode
  </td></tr>
<tr><td colspan="3">1 * 面包板
  </td></tr>
<tr><td>1 * USB 线
  </td><td colspan="2">一些跳线
  </td></tr>
</table>

- *SunFounder Mega 板*
- 面包板
- 跳线
- 电阻
- *7 段数码管*

7.17.3 原理图

在本实验中，将 7 段数码管的每个引脚 a~g 分别连接到一个 220 欧姆的限流电阻，然后连接到引脚 4-11。GND 连接到 GND。通过编程，我们可以将 pin4-11 中的一个或几个设置为高电平来点亮相应的 LED。

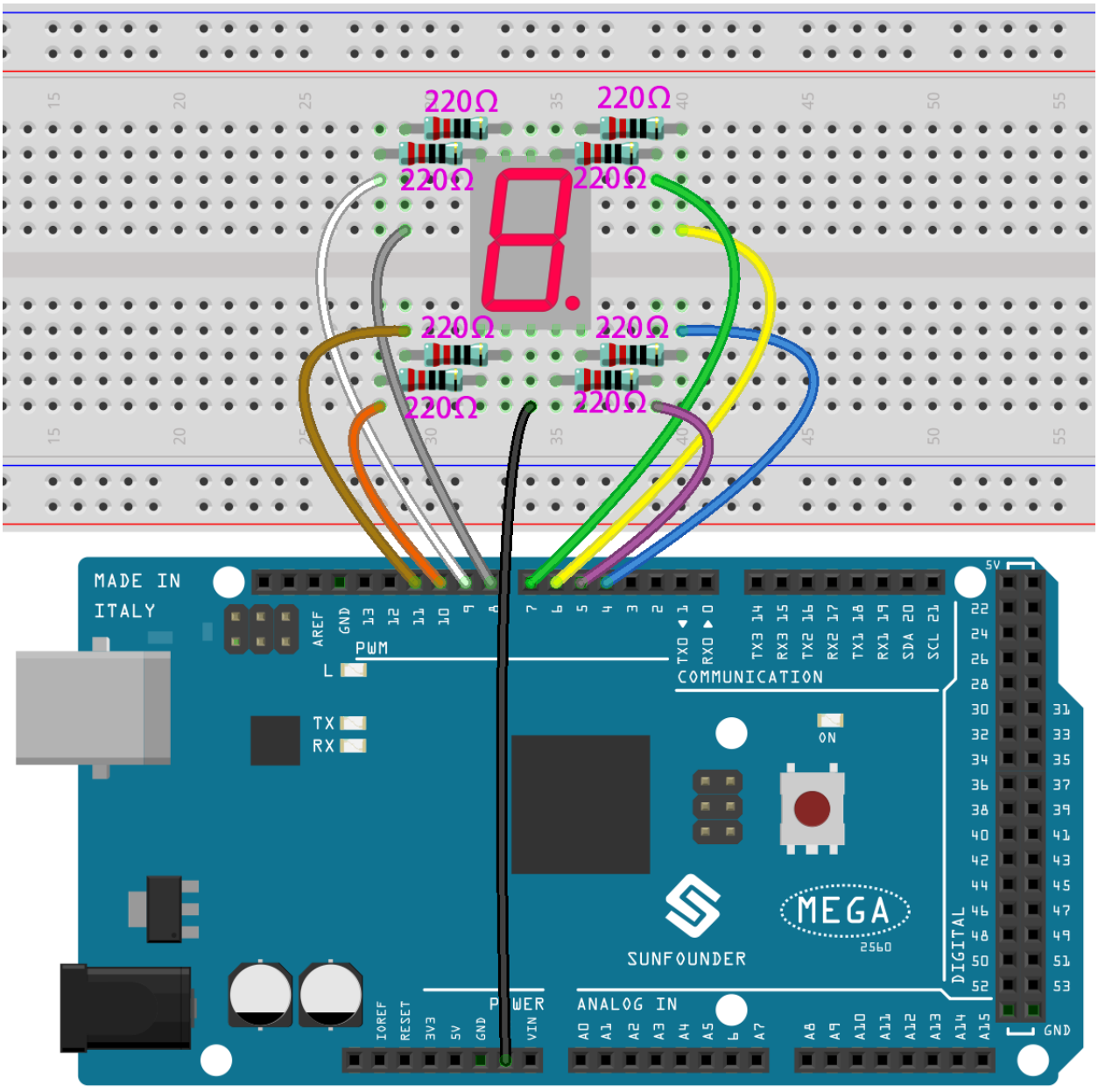


7.17.4 实验步骤

第 1 步：搭建电路 (这里使用的是共阴极 7 段数码管)。

7 段数码管与 Mega2560 板的接线：

7 段数码管	Mega 板
a	7
b	6
c	5
d	11
e	10
f	8
g	9
dp	4
“- “	GND

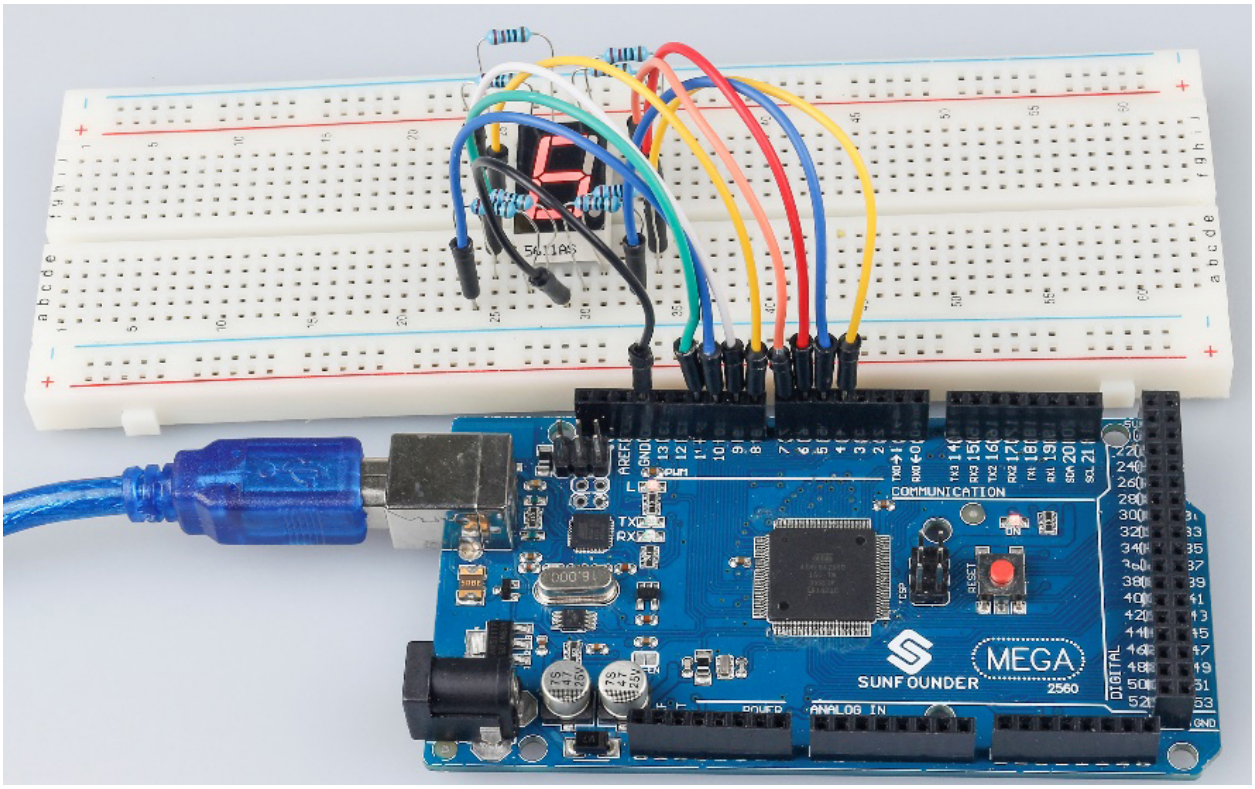


第 2 步：打开代码文件 `Lesson_17_7_Segment_Display.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

你现在应该看到 7 段数码管轮流显示 0-9，A-F。



7.17.5 代码

7.17.6 代码分析

这个实验的代码可能有点长。但是语法很简单。让我们来看看。

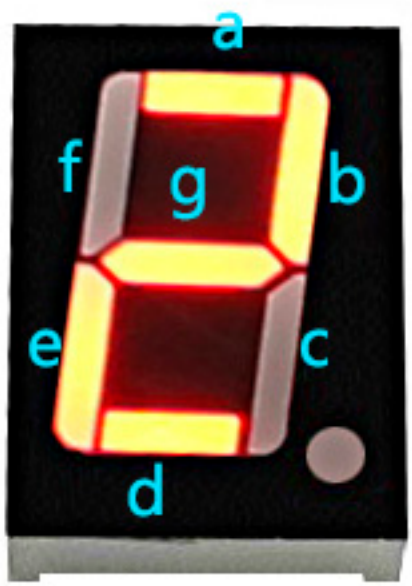
在 loop() 中调用函数

```
void loop()
{
    digital_1();//diaplay 1 to the 7-segment
    delay(1000);//wait for a second
    digital_2();//diaplay 2 to the 7-segment
    delay(1000); //wait for a second
    digital_3();//diaplay 3 to the 7-segment
    delay(1000); //wait for a second
    digital_4();//diaplay 4 to the 7-segment
...
```

将这些函数调用到 loop() 中是为了让 7 段数码管显示 0-F。功能如下所示。以 `digital_2()` 为例：

digital_2() 详解

```
void digital_2(void) //diaplay 2 to the 7-segment
{
digitalWrite(b,HIGH);
digitalWrite(a,HIGH);
for(int j = 9;j <= 11;j++)
    digitalWrite(j,HIGH);
digitalWrite(c,LOW);
digitalWrite(f,LOW);
}
```



首先我们需要知道在 7 段数码管上显示数字 2 时的样子。实际上是 a、b、d、e 和 g 段通电（被设置为高电平），c 和 f 熄灭（被设置为高电平），从而产生 2 的显示。

运行此部分后，7 段数码管将显示 2。同样，其他字符的显示也是一样的。由于大写字母 b 和 d ，即 B 和 D，在数码管上与 8 和 0 看起来相同，因此它们以小写字母显示。

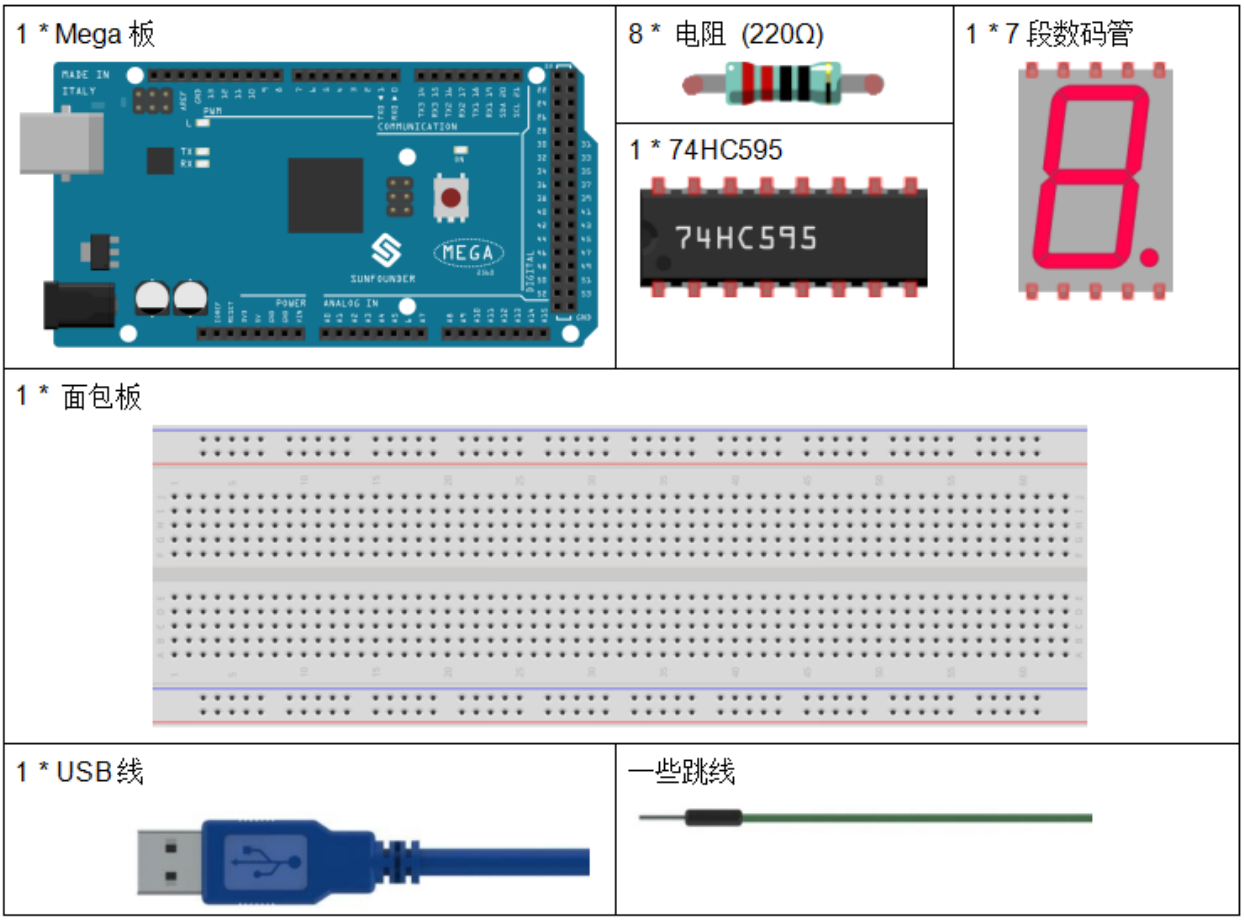
7.18 第 18 课 74HC595

7.18.1 介绍

通常，有两种方式可以驱动单个 7 段数码管。一种方法是将其 8 个引脚直接连接到主板上的 8 个端口，我们之前已经这样做了。或者你可以将 74HC595 连接到主板的三个端口，然后将 7 段数码管连接到 74HC595。

在本实验中，我们将使用后者。这样，我们可以节省五个端口——考虑到主板的端口有限，这一点非常重要。现在让我们开始吧！

7.18.2 所需器件



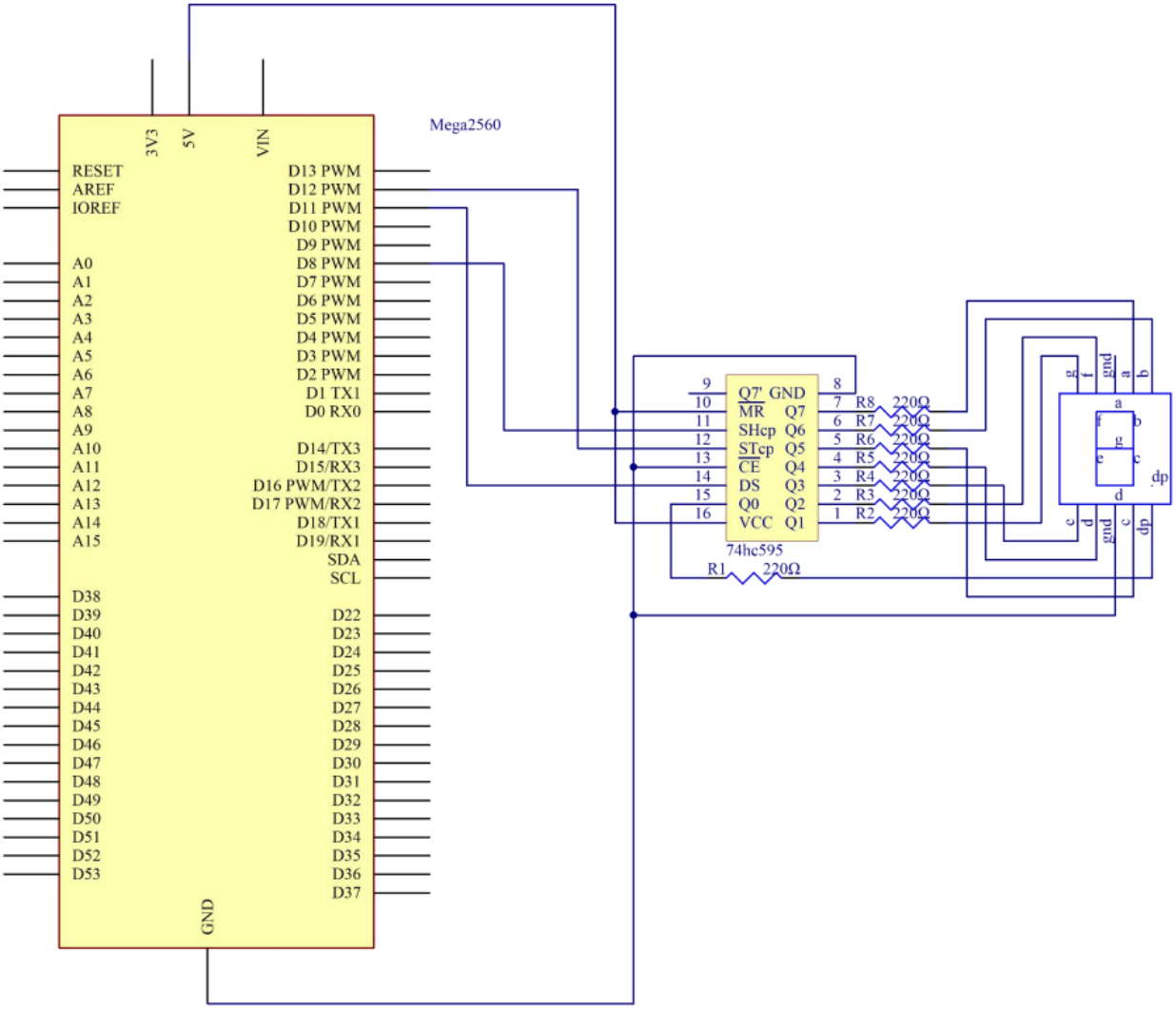
- *SunFounder Mega 板*
- *面包板*
- *跳线*
- *电阻*
- *7 段数码管*
- *74HC595*

7.18.3 原理图

在实验中，MR（引脚 10）连接到 5V（高电平），OE（引脚 1）连接到 GND（低电平）。因此，数据在 SHcp 的上升沿输入，通过上升沿进入内存寄存器。我们使用 `shiftout()` 函数通过 DS 输出一个 8 位的数据到移位寄存器。在 SHcp 的上升沿，移位寄存器中的数据一次连续移动一位，即 Q1 中的数据移动到 Q2，以此类推。在 STcp 的上升沿，移位寄存器中的数据移动到内存寄存器中。

所有数据将在 8 次后移动到内存寄存器。然后将内存寄存器中的数据输出到总线（Q0-Q7）。所以 16 个字符依次显示在 7 段数码管中。

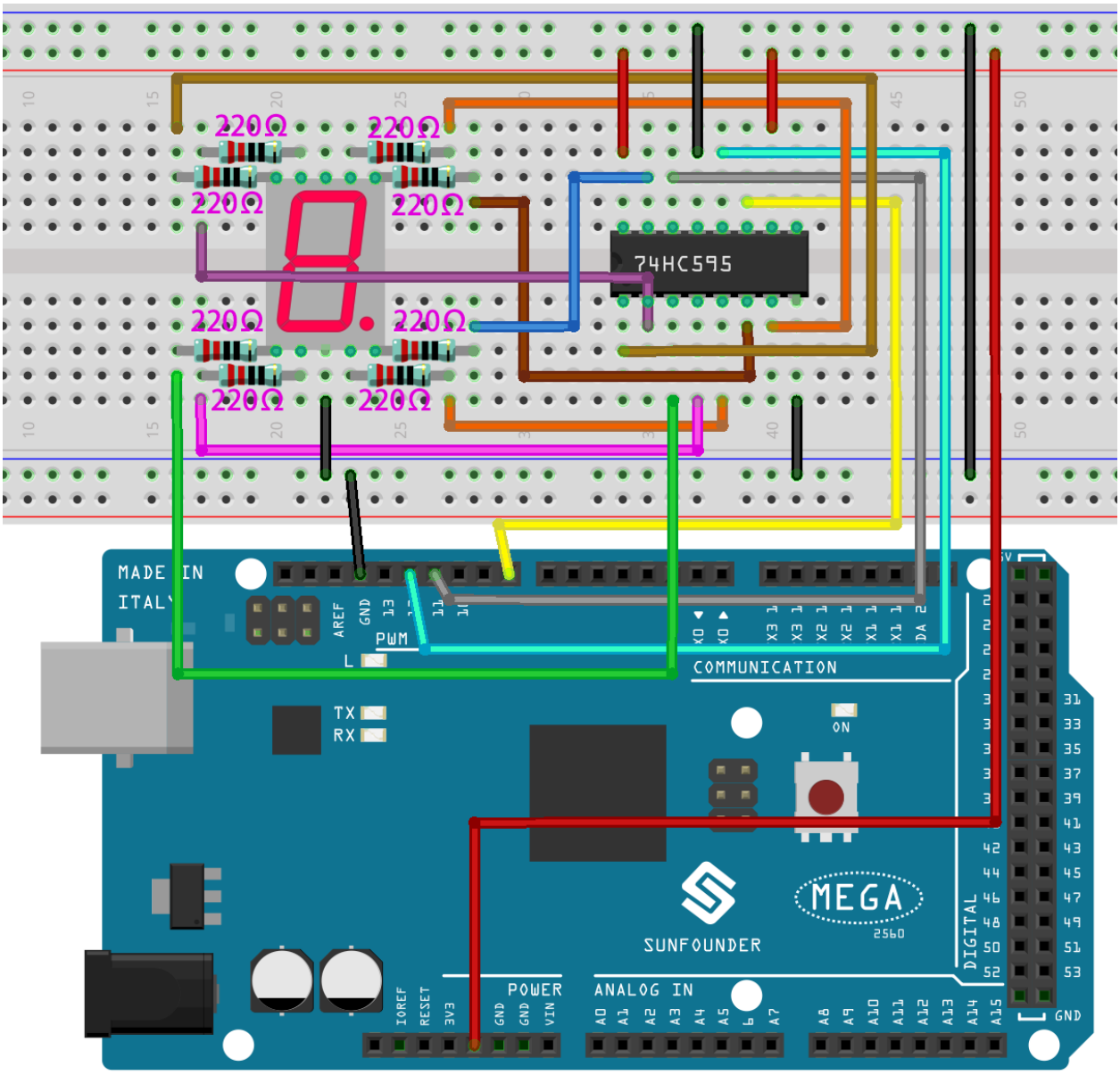
原理图如下所示：



7.18.4 实验步骤

第 1 步：搭建电路。(注意芯片的槽口方向，应该是在左侧)

7 段数码管	74HC595	Mega 板
a	Q7	
b	Q6	
c	Q5	
d	Q4	
e	Q3	
f	Q2	
g	Q1	
DP	Q0 VCC DS CE ST SH MR Q7' GND	5V 11 GND 12 8 5V N/C GND
•		GND

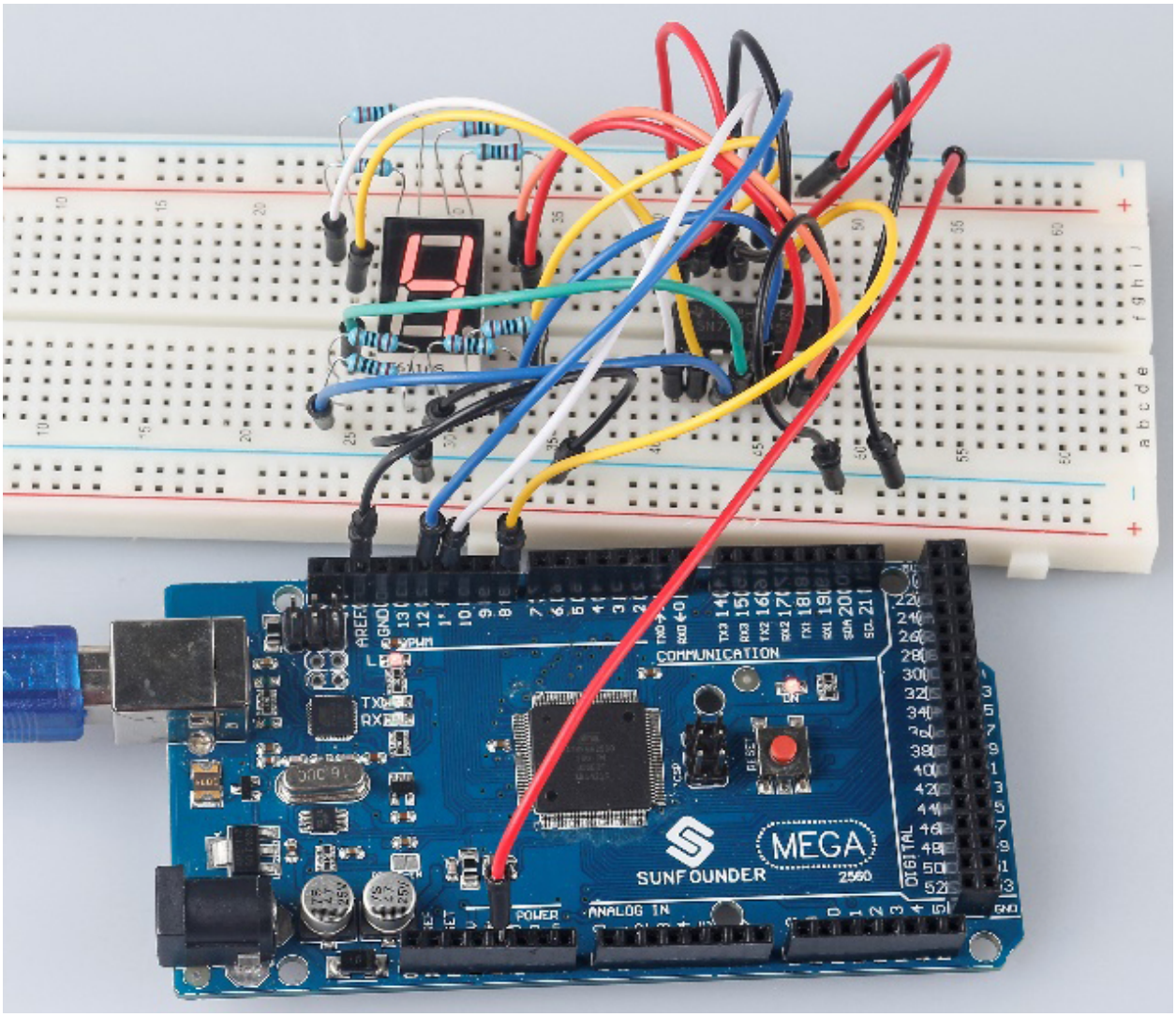


第 2 步：打开代码文件 `Lesson_18_74HC595.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

你将看到 7 段数码管显示 0-9，A-F。



7.18.5 代码

7.18.6 代码分析

设置数组和元素

```
int datArray[16] = {252, 96, 218, 242, 102, 182, 190, 224, 254, 246, 238, 62, 156,␣
↪122, 158, 142};
```

这个数组存放了从 0 到 F 的 16 个字符的数据，252 代表 0，可以自己计算。要显示 0，7 段数码管的 g 段（中间的）必须是低电平（暗）。

由于 g 连接到 74HC595 的 Q1，将 Q1 和 DP（点）都设置为低电平，其余引脚为高电平。因此，Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 的值为 1 1 1 1 1 1 0 0。

将二进制数改为十进制数：1x2^7 + 1x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 =252

这就是要显示的数字 0 的值。你可以类似地计算其他字符。

在 7 段数码管上显示 0-F

```
for(int num = 0; num < 16; num++)
{
    digitalWrite(STcp,LOW); //ground ST_CP and hold low for as long as you are␣
↪transmitting
    shiftOut(DS,SHcp,MSBFIRST,datArray[num]);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(STcp,HIGH); //pull the ST_CPST_CP to save the data
    delay(1000); //wait for a second
}
```

将 STcp 设置为低电平，然后设置为高电平。它将产生一个的上升沿脉冲。

shiftOut() 用于逐位移出一个字节的数据，即将 dataArray[num] 中的一个字节数据移到 DS 引脚的移位寄存器中。MSBFIRST 表示从高位移动。之后 digitalWrite（STCP，HIGH）运行时，STCP 将在上升沿。这时，移位寄存器中的数据就会被移到内存寄存器中。

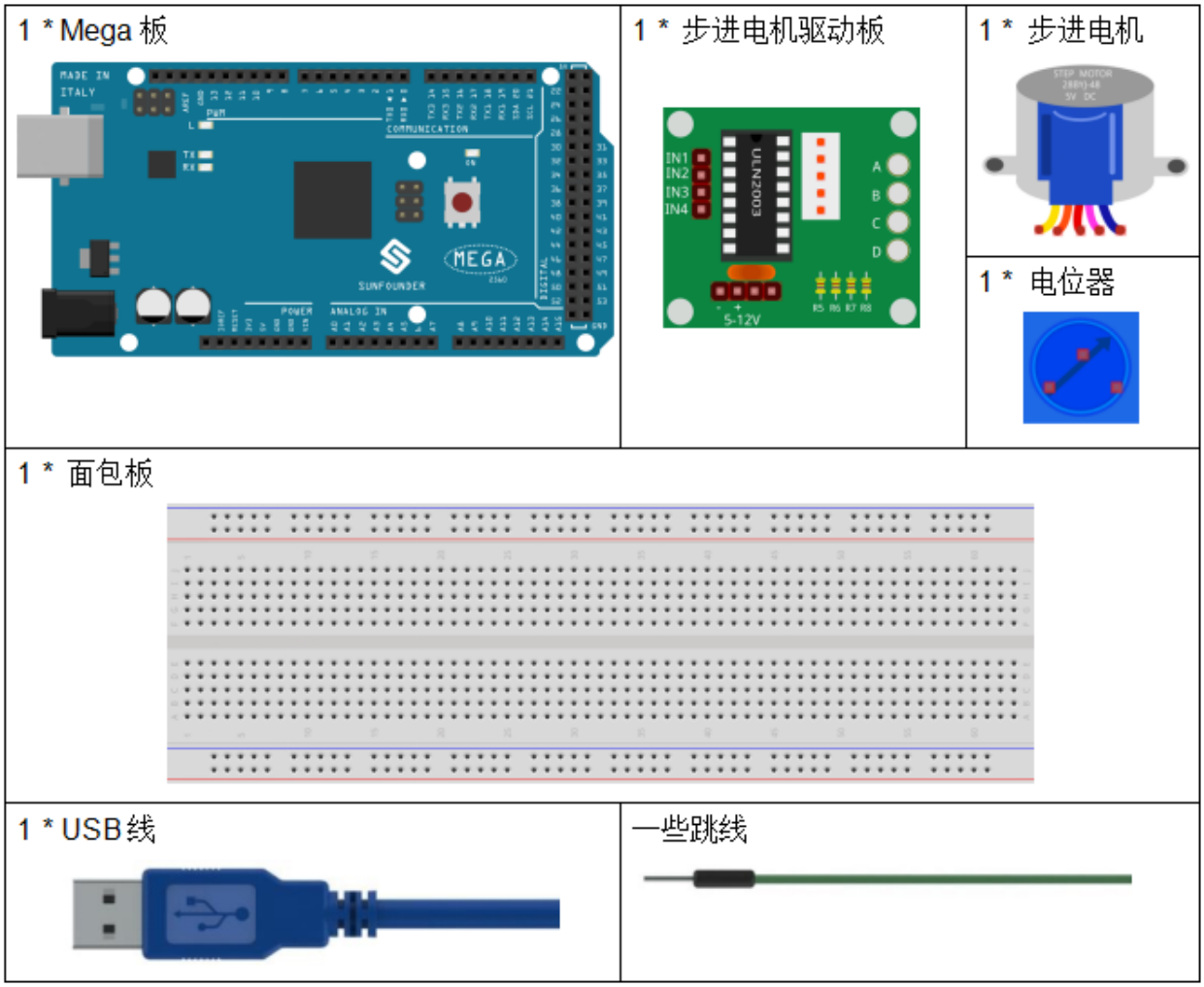
8 次后，一个字节的数据将被传送到内存寄存器中。然后将内存寄存器的数据输出到总线（Q0-Q7）。你将看到一个字符显示在 7 段数码管上，然后延迟 1000ms。在该行之后，返回 for()。如此循环直到 16 次后，7 段数码管中的所有字符都一一显示出来。

7.19 第 19 课步进电机

7.19.1 介绍

步进电机由于其独特的设计，可以在没有任何反馈机制的情况下进行高精度控制。步进电机的轴上装有一系列磁铁，由一系列电磁线圈控制，这些线圈按特定顺序带正负电，以小“步”精确地向前或向后移动。

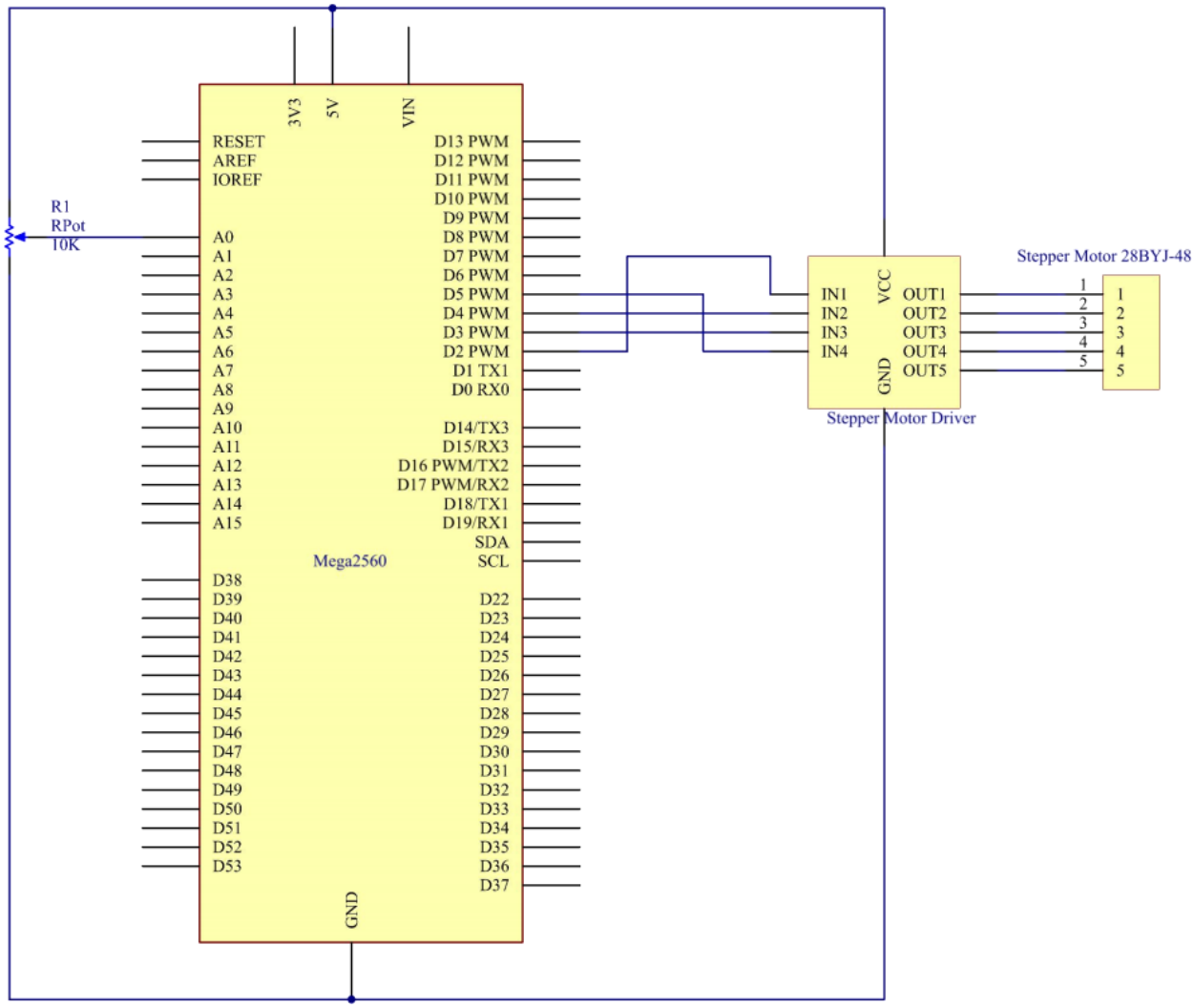
7.19.2 所需器件



- *SunFounder Mega 板*
- 面包板
- 跳线
- 电位器
- 步进电机

7.19.3 原理图

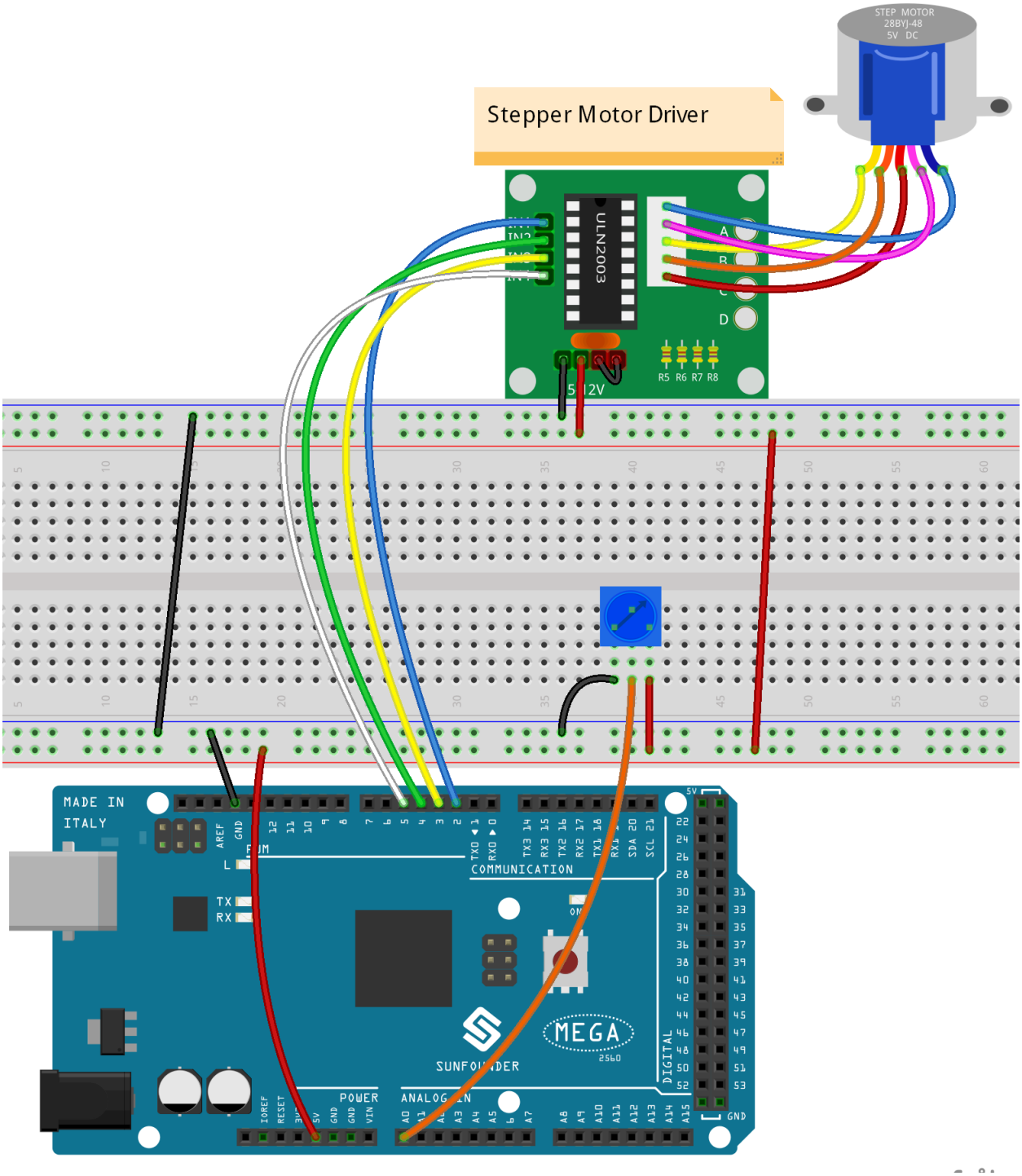
原理图如下所示：



7.19.4 实验步骤

第 1 步：搭建电路。

步进电机驱动板	Mega 板
IN1	2
IN2	4
IN3	3
IN4	5
GND	GND
VCC	5v

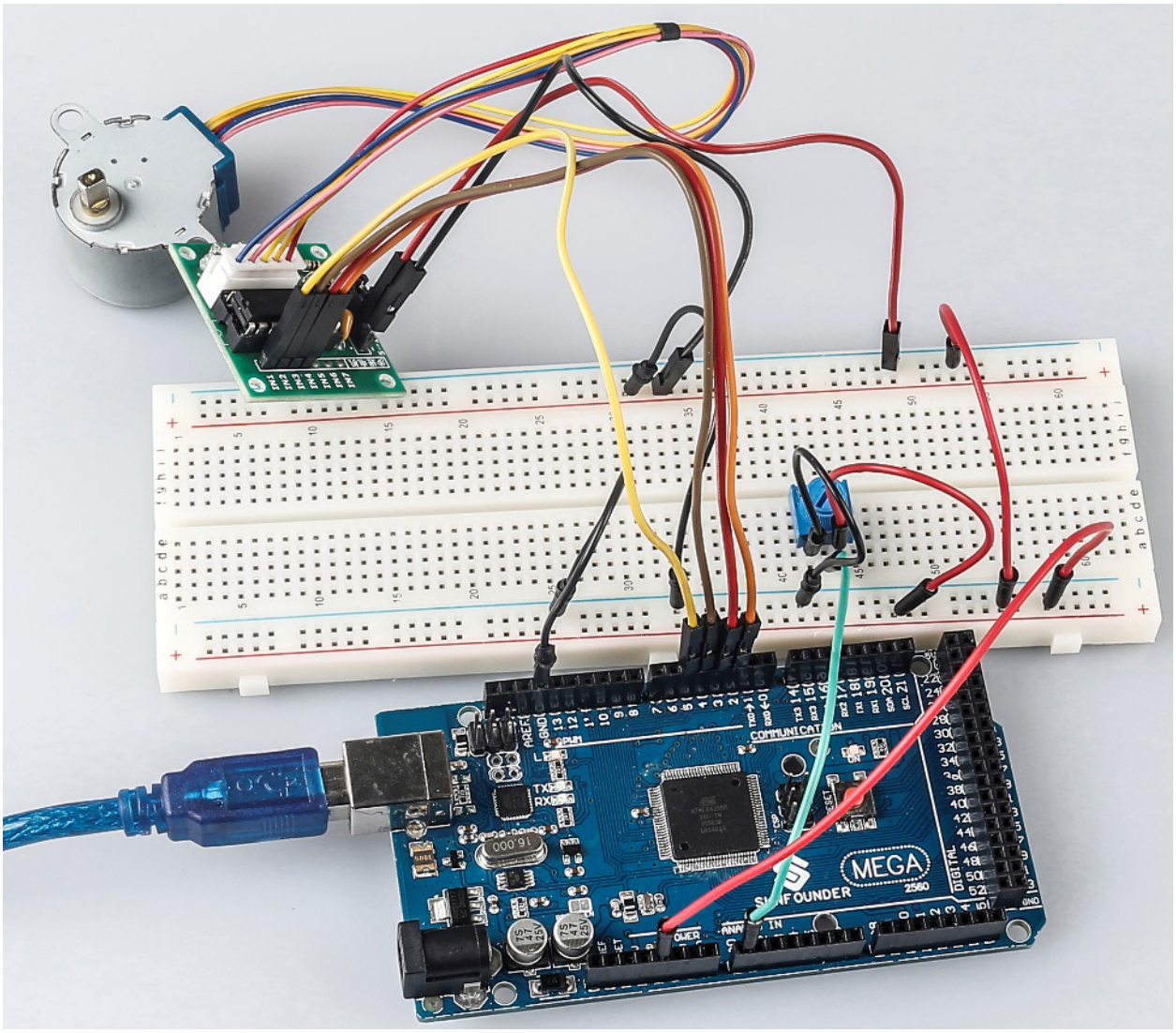


第 2 步：打开代码文件 `Lesson_19_Stepper_Motor.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你应该看到步进电机的摇臂顺时针和逆时针交替旋转。



7.19.5 代码

7.19.6 代码分析

初始化步进电机

```
#include <Stepper.h> //include a head file
//the steps of a circle
#define STEPS 2048
//set steps and the connection with MCU
Stepper stepper(STEPS, 2, 3, 4, 5);
//available to store previous value
int previous = 0;
```

包含头文件 `Stepper.h`，将步长设置为 100，然后使用函数 `stepper()` 初始化步进电机。

- `Stepper(steps, pin1, pin2, pin3, pin4)`：此函数创建 `Stepper` 类的新实例，代表连接到 Arduino 板的特定步进电机。

- `steps`：电机旋转一圈的步数。如果你的电机给出每步的度数，将该数字除以 360 以获得步数（例如 360 / 3.6 给出 100 步，整数型）。

setSpeed() 函数

```
//speed of per minute
stepper.setSpeed(15); //set the motor speed in rotations per minute(RPMs)
```

- `setSpeed(rpms)`：以每分钟转数 (RPMs) 为单位设置电机速度。此函数不会使电机转动，只是设置调用 `step()` 时的速度。
- `rpms`：电机每分钟旋转的速度 - 一个正数（长型）。

主程序

```
void loop()
{
//get analog value
int val = analogRead(0); //Read the value of the potentiometer
//current reading minus the reading of history
stepper.step(val - previous); //Turn the motor in val-previous steps
//store as prevous value
previous = val; //the value of potentiometer assignment to variable previous
}
```

主程序是先读取 A0 的值，然后根据 A0 的值来设置步进电机转动的步数。

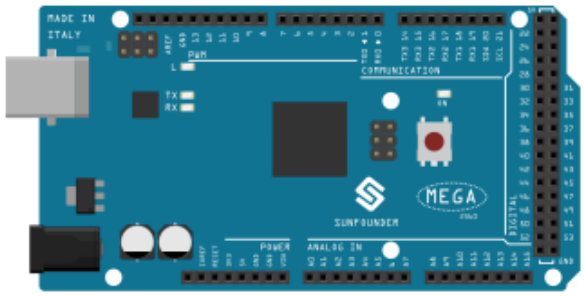
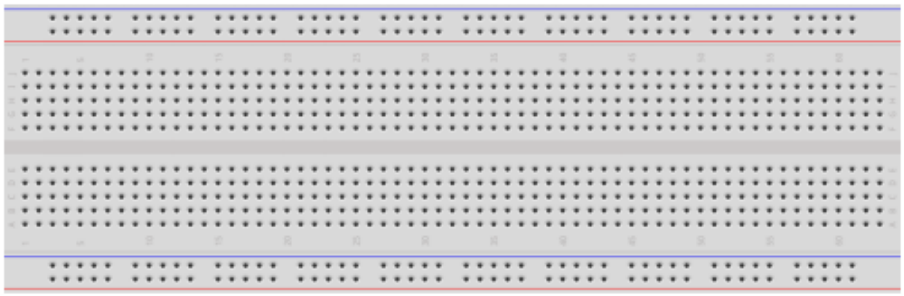
- `step(steps)`：以特定的步数转动电机，速度由最近调用 `setSpeed()` 确定。这个功能是阻塞的；也就是说，它将等到电机完成移动后才能将控制权传递给代码中的下一行。例如，如果你将速度设置为 1 RPM 并在 100 步电机上调用 `step(100)`，则此函数将需要整整一分钟才能运行。为了更好地控制，保持高速并且每次调用 `step()` 时只走几步。
- `steps`：转动电机的步数 - 正向转动一个方向，负向转动另一个（int）。

7.20 第 20 课简单创作 - 秒表

7.20.1 介绍

在这个课程中，我们将使用 4 位 7 段数码管来制作一个秒表。

7.20.2 所需器件

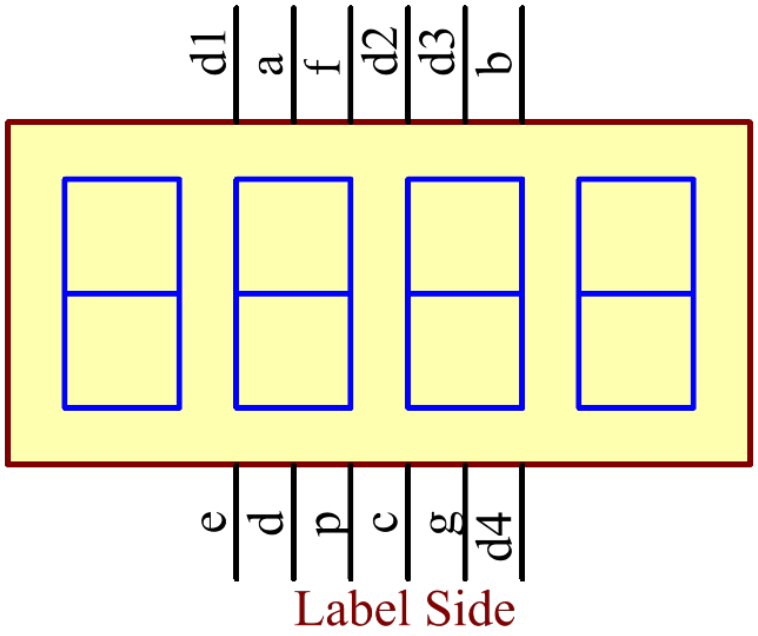
<table>
<tr><td>1 * Mega 板
  </td><td>8 * 电阻 (220Ω)
  </td><td>1 * 4 位 7 段数码管
  </td></tr>
<tr><td colspan="3">1 * 面包板
  </td></tr>
<tr><td>1 * USB 线
  </td><td colspan="2">一些跳线
  </td></tr>
</table>

- *SunFounder Mega 板*
- *面包板*
- *跳线*
- *电阻*
- *4 位 7 段数码管*

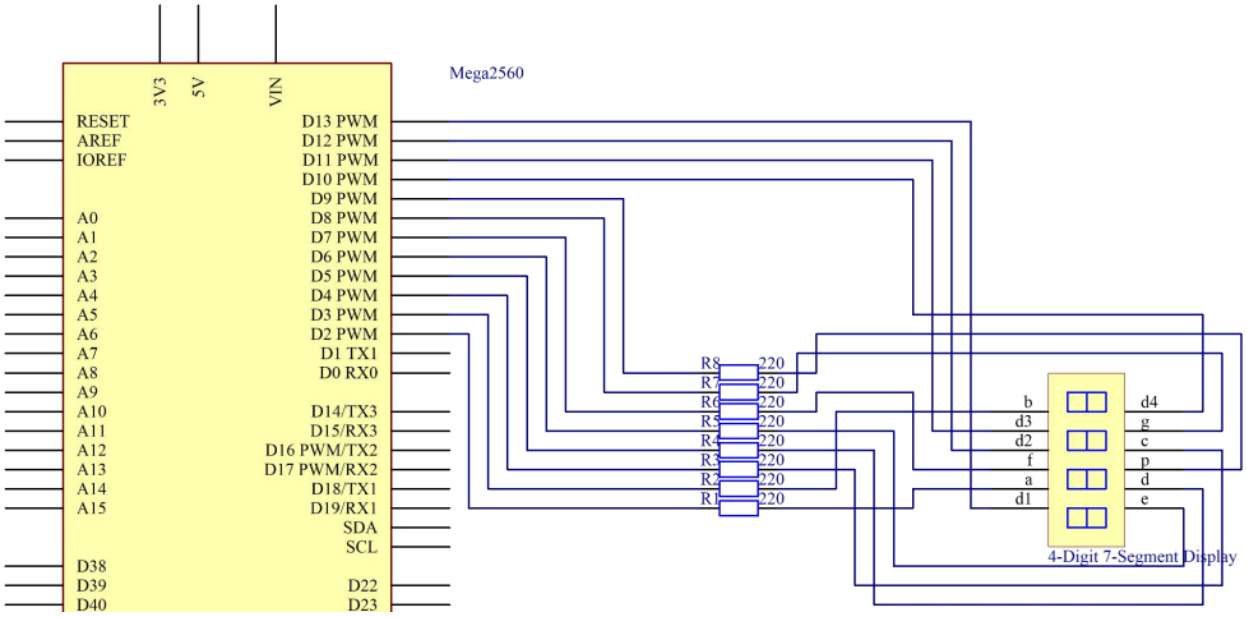
7.20.3 原理图

使用 7 段数码管时，如果是共阳极数码管，将阳极引脚接电源；如果是共阴极，则将阴极引脚连接到 GND。使用 4 位 7 段数码管时，共阳极或共阴极管脚控制显示的数字。只有一位数字有效，但是，根据视觉暂留原理，我们可以看到四个 7 段数码管都显示数字。这是因为电子扫描速度太快，我们无法注意到间隔。

4 位 7 段数码管示意图如下：


d1
a
f
d2
d3
b
e
d
p
c
g
d4
Label Side

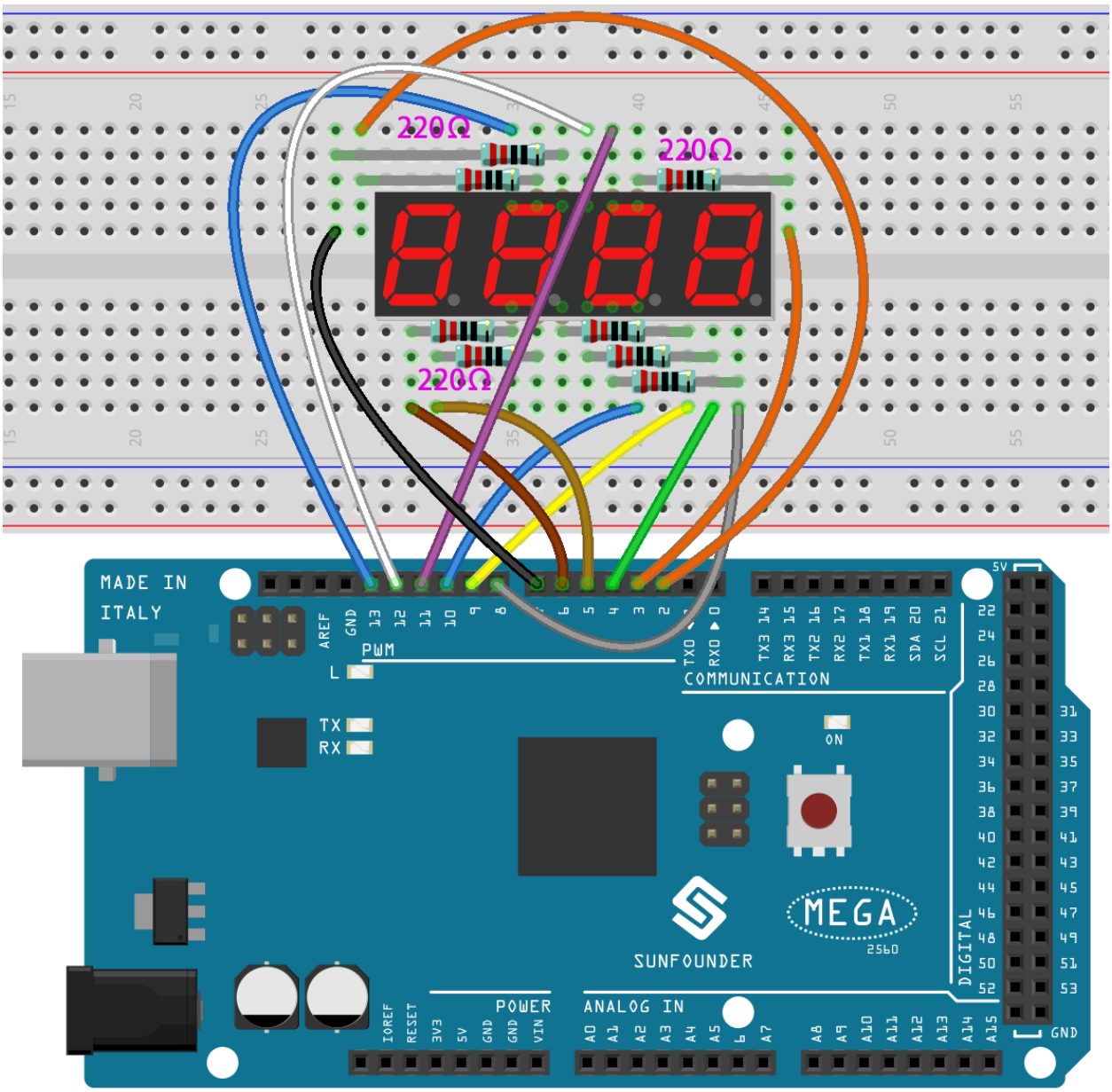
原理图如下所示：


Mega2560
3V3
5V
VIN
RESET
AREF
IOREF
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
D38
D39
D40
D13 PWM
D12 PWM
D11 PWM
D10 PWM
D9 PWM
D8 PWM
D7 PWM
D6 PWM
D5 PWM
D4 PWM
D3 PWM
D2 PWM
D1 TX1
D0 RX0
D14/TX3
D15/RX3
D16 PWM/TX2
D17 PWM/RX2
D18/TX1
D19/RX1
SDA
SCL
D22
D23
R8 220
R7 220
R6 220
R5 220
R4 220
R3 220
R2 220
R1 220
b
d3
d2
f
a
d1
d4
g
c
p
d
e
4-Digit 7-Segment Display

7.20.4 实验步骤

第 1 步：搭建电路。

4 位 7 段数码管	Mega 板
a	2
b	3
c	4
d	5
e	6
f	7
g	8
p	9
D1	13
D2	12
D3	11
D4	10

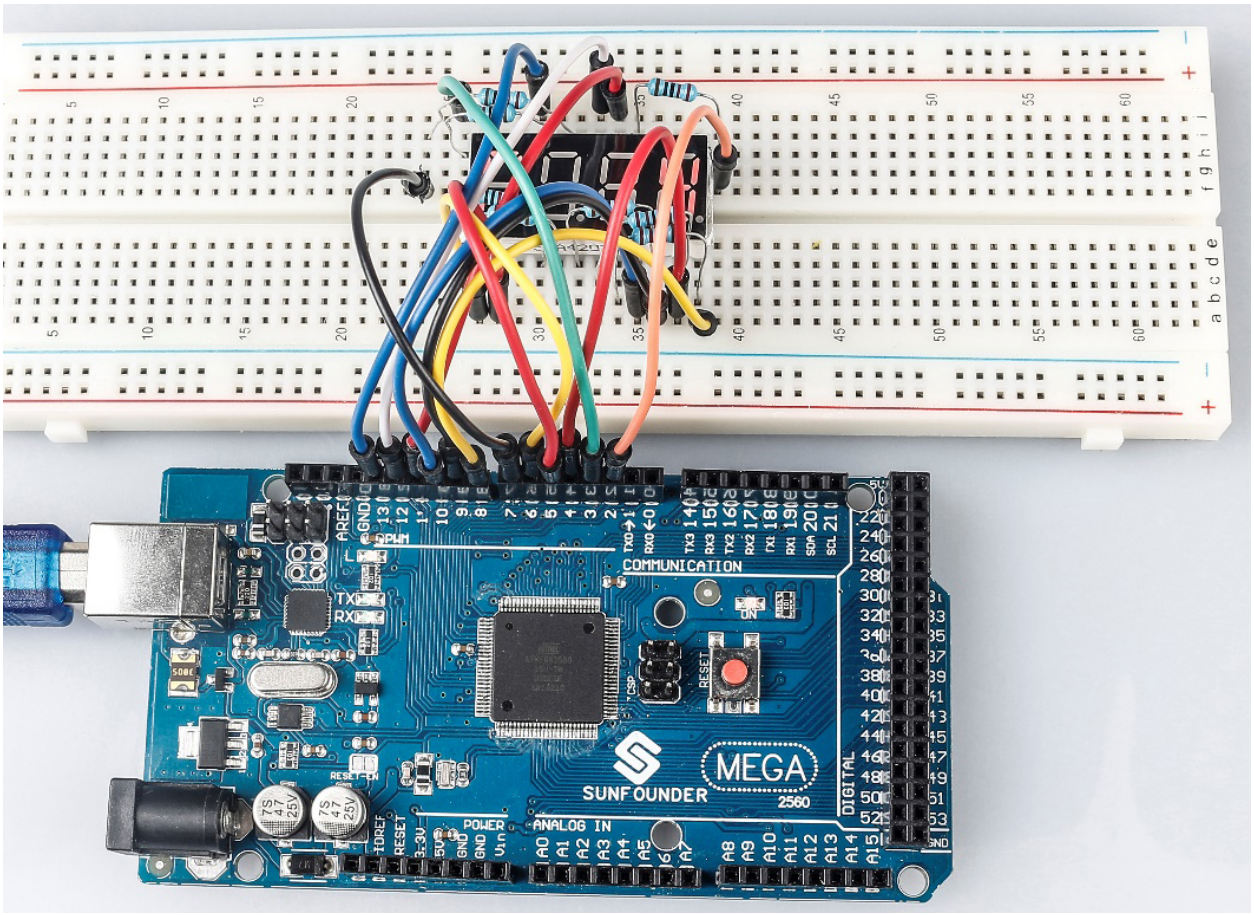


第 2 步：打开代码文件 `Lesson_20_Stopwatch.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

现在，你可以在 4 位 7 段数码管上看到数字每秒增加 1。



7.20.5 代码

7.20.6 代码分析

这就是代码的全部内容，比较长，我总结一下：

将 4 位 7 段数码管的所有引脚设置为输出。设置定时器 1 为 0.1 秒，所以当 0.1 秒的时候，`add()` 会被调用；但是在 0.1 秒过去之前，`add()` 还没有被调用。然后运行 `loop()` 函数，4 个数码管显示为 0000。等待一段时间，0.1 秒后，表明 count=10，调用函数 `add()`。然后 n++=1；因为 1<10000，不会恢复到 0。运行 `loop()`，数码管显示为 0001。0.1 秒后，n 增加 1，n++=2，显示将变成 0002，然后是 0003，一直到 9999。n 每秒增加 1，显示的数字也相应增加，直到 n=10000，n 再次为 0。然后从 0 开始计数。

初始化定时器

```
Timer1.initialize(100000);
// set a timer of length 100000 microseconds(or 0.1 sec - or 10Hz => the led will␣
↪blink 5 times, 5 cycles of on-and-off, per second)

Timer1.attachInterrupt( add ); // attach the service routine here
```

语句 `attachInterrupt(add)` 就是附加一个 ISR 函数，当有中断时调用 `add()` 函数。

Loop 函数

```
void loop()
{
    clearLEDs();//clear the 7-segment display screen
    pickDigit(0);//Light up 7-segment display d1
    pickNumber((n / 1000)); // get the value of thousand
    delay(del);//delay 5ms

    clearLEDs();//clear the  7-segment display screen
    pickDigit(1);//Light up 7-segment display d2
    pickNumber((n % 1000) / 100); // get the value of hundred
    delay(del);//delay 5ms

    clearLEDs();//clear the  7-segment display screen
    pickDigit(2);//Light up 7-segment display d3
    pickNumber(n % 100 / 10); //get the value of ten
    delay(del);//delay 5ms

    clearLEDs();//clear the 7-segment display screen
    pickDigit(3);//Light up 7-segment display d4
    pickNumber(n % 10); //Get the value of single digit
    delay(del);//delay 5ms
}
```

loop() 用于让四段显示器显示一个数值的个位数、十位、十万位。

如 n=1345、(1345/1000)=1、(1345%1000)/100)=3、((1345%100)/10)=4、(n%10)=5

pickDigit(int x) 函数

```
void pickDigit(int x)  //light up a 7-segment display
{
    //The 7-segment LED display is a common-cathode one. So also use digitalWrite to
↪set d1 as high and the LED will go out
    digitalWrite(d1, HIGH);
    digitalWrite(d2, HIGH);
    digitalWrite(d3, HIGH);
    digitalWrite(d4, HIGH);

    switch (x)
    {
        case 0:
            digitalWrite(d1, LOW);//Light d1 up
            break;
        case 1:
            digitalWrite(d2, LOW); //Light d2 up
            break;
        case 2:
            digitalWrite(d3, LOW); //Light d3 up
            break;
        default:
            digitalWrite(d4, LOW); //Light d4 up
            break;
    }
}
```

4 位 7 段数码管为共阴的，将 d1、d2、d3、d4 全部设置为 HIGH 使其熄灭。

再来判断 x 的值：

- x 为 0，让 d1 为低电平来让第 4 个数码管（左边第一个）工作。
- x 为 1，让第 3 个数码管工作。
- x 为 2，让第 2 个数码管工作。
- 默认情况下，让第 1 个数码管（右边第一个）工作。

pickNumber(int x) 函数

```
void pickNumber(int x)
{
    switch (x)
    {
        default:
            zero();
            break;
        case 1:
            one();
            break;
        case 2:
            two();
            break;
        case 3:
            three();
            break;
        ...
```

这个函数的功能是控制 LED 显示数字。调用 `zero()`、`one()` 直到 `nine()` 函数显示 0-9 数字。

通过 x 的值来判断显示什么数字：

- 默认情况，调用 `zero()` 函数来显示 0。
- x 为 1，调用 `one()` 函数来显示 1。
- x 为 2，调用 `two()` 函数来显示 2。
- x 为 3，调用 `three()` 函数来显示 3。
- x 为 4，调用 `four()` 函数来显示 4。
- x 为 5，调用 `five()` 函数来显示 5。
- x 为 6，调用 `six()` 函数来显示 6。
- x 为 7，调用 `seven()` 函数来显示 7。
- x 为 8，调用 `eight()` 函数来显示 8。
- x 为 9，调用 `nine()` 函数来显示 9。

以 `zero()` 为例：

`zero()` 函数是控制 LED 的高低电平。使用 `digitalWrite()` 将 a 设置为 f 为高，g 为低。根据刚才提到的引脚图，当 a 到 f 为高，g 为低时，会显示数字 0。

```
void zero() //the  7-segment led display 0
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
```

(续下页)

(接上页)

```
    digitalWrite(f, HIGH);
    digitalWrite(g, LOW);
}
```

clearLEDs() 函数

```
void clearLEDs() //clear the  7-segment display screen
{
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(p, LOW);
}
```

将 a-p 引脚都设置为低电平来让 4 位 7 段数码管全部熄灭。

add() 函数

```
void add()
{
    // Toggle LED
    count ++;
    if(count == 10)
    {
    count = 0;
    n ++;
    if(n == 10000)
    {
        n = 0;
    }
    }
}
```

`count` 的初始值是 0，将 `count` 累加；加到 10 再重置为 0，此时将 `n` 累加；`n` 加到 10000 后，再重置为 0。

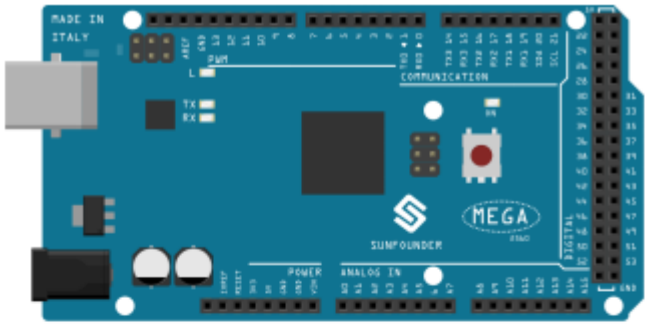
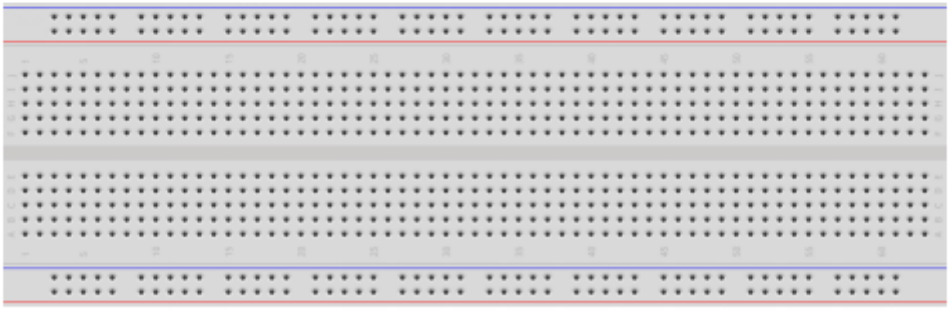
7.21 第 21 课简单创作 - 抢答器

7.21.1 介绍

在问答比赛中，尤其是娱乐活动（如竞技答题活动）中，主办方为了准确、公正、直观地确定答题者的座位号，往往会采用蜂鸣器系统。

现在系统可以用数据来说明判断的准确性和公平性，提高了娱乐性。同时，也更加公平公正。在本课中，我们将使用一些按键、蜂鸣器和 LED 来制作测验蜂鸣器系统。

7.21.2 所需器件

<table>
<tr><td colspan="2">1 * Mega 板
  </td><td>4 * 电阻 (220Ω)
  </td><td rowspan="2">4 * 按键
  </td></tr>
<tr><td colspan="2"></td><td>1 * 有源蜂鸣器
  </td></tr>
<tr><td>1 * 红色 LED
  </td><td>1 * 黄色 LED
  </td><td>1 * 绿色 LED
  </td><td>1 * 蓝色 LED
  </td></tr>
<tr><td colspan="4">1 * 面包板
  </td></tr>
<tr><td colspan="2">1 * USB 线
  </td><td colspan="2">一些跳线
  </td></tr>
</table>

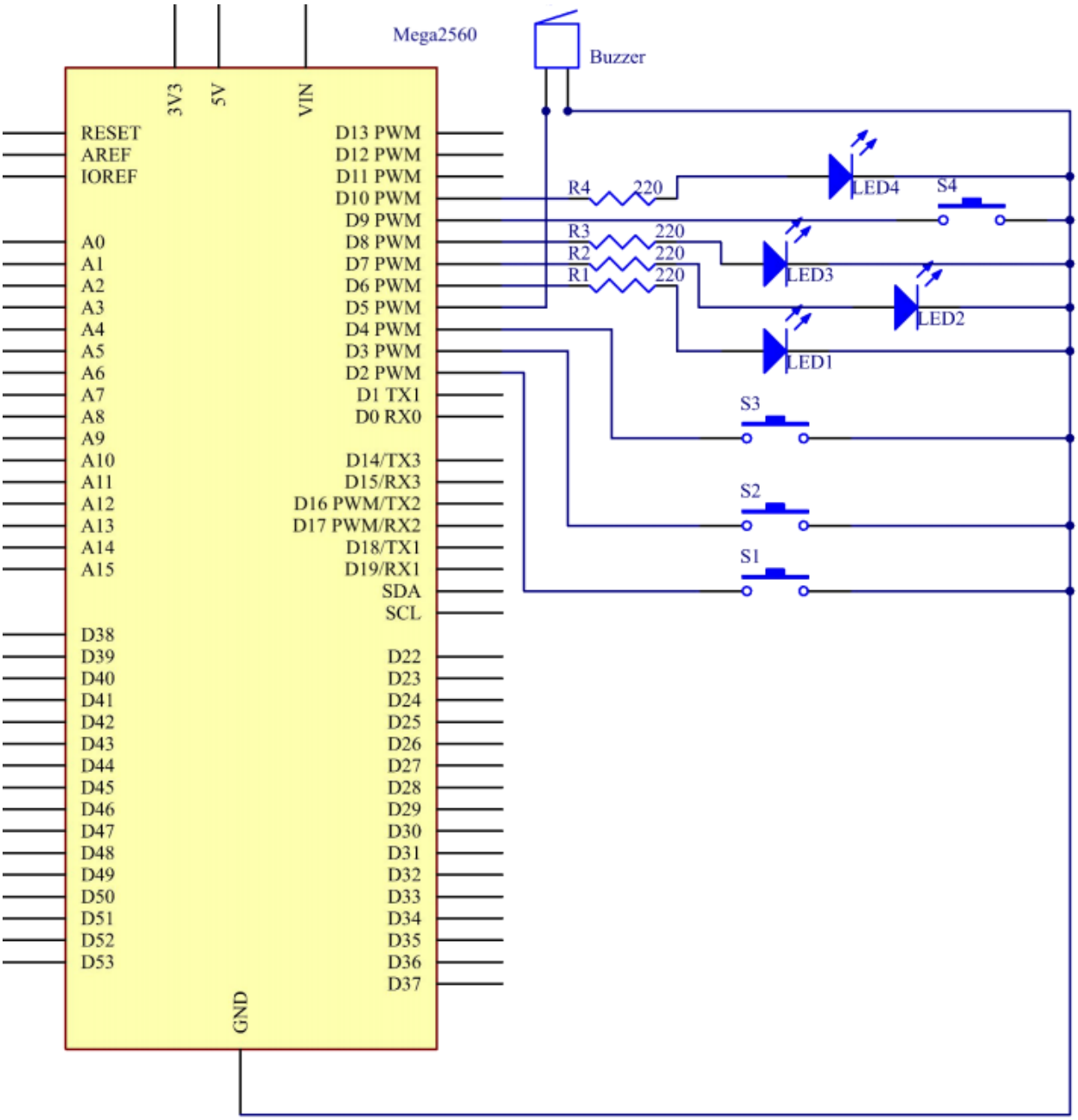
- *SunFounder Mega 板*

- 面包板
- 跳线
- 电阻
- *LED* 发光二极管
- 按键
- 蜂鸣器

7.21.3 原理图

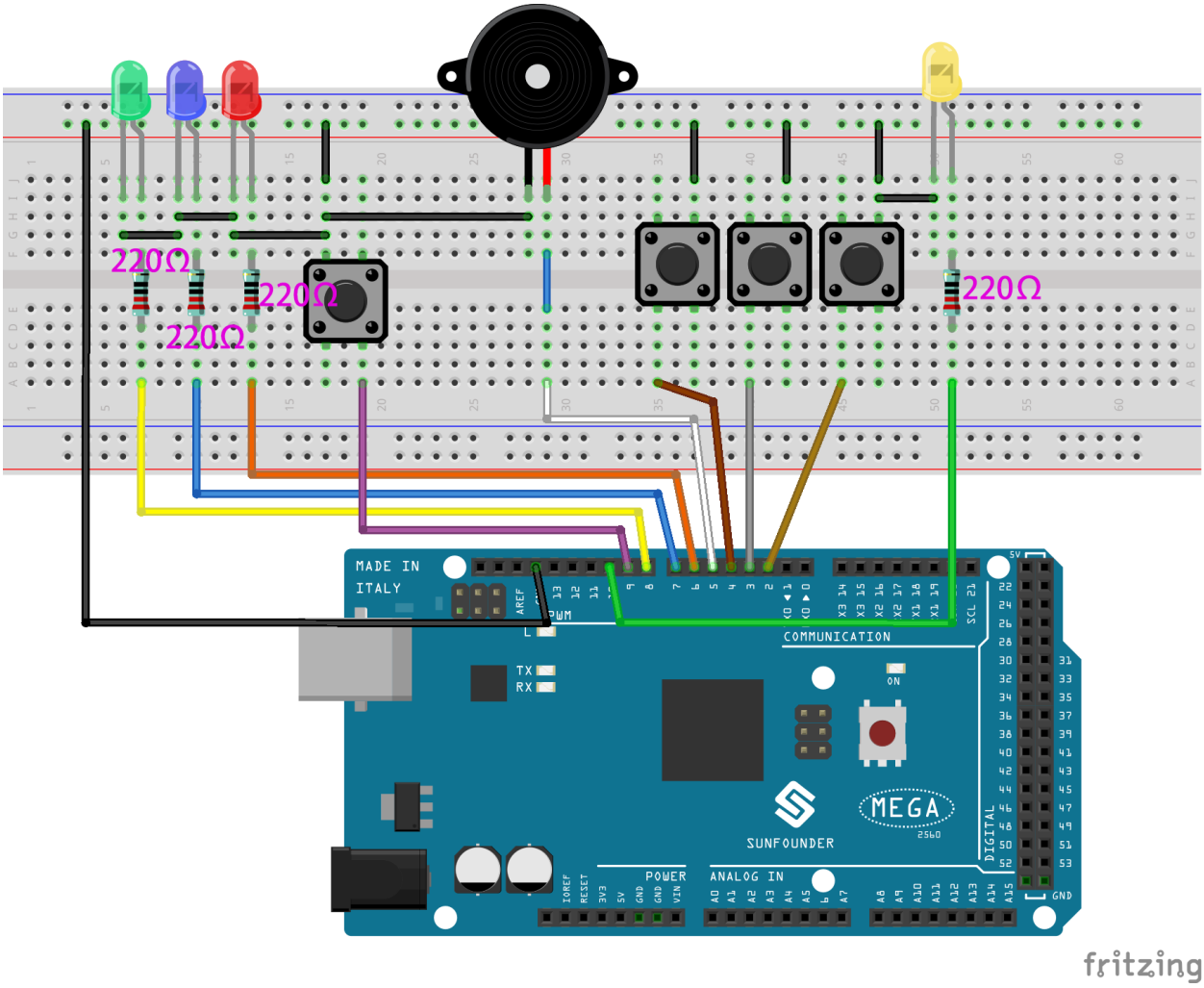
按键 1、2 和 3 是抢答按键，按键 4 是重置按键。如果先按下按键 1，蜂鸣器将发出蜂鸣声，相应的 LED 将亮起，所有其他 LED 将熄灭。如果要开始另一轮，请按按键 4 重置。

原理图如下所示：


Mega2560
Buzzer
3V3
5V
VIN
RESET
AREF
IOREF
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
D38
D39
D40
D41
D42
D43
D44
D45
D46
D47
D48
D49
D50
D51
D52
D53
GND
D13 PWM
D12 PWM
D11 PWM
D10 PWM
D9 PWM
D8 PWM
D7 PWM
D6 PWM
D5 PWM
D4 PWM
D3 PWM
D2 PWM
D1 TX1
D0 RX0
D14/TX3
D15/RX3
D16 PWM/TX2
D17 PWM/RX2
D18/TX1
D19/RX1
SDA
SCL
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
D32
D33
D34
D35
D36
D37
R4 220
R3 220
R2 220
R1 220
LED4
LED3
LED2
LED1
S4
S3
S2
S1

7.21.4 实验步骤

第 1 步：搭建电路

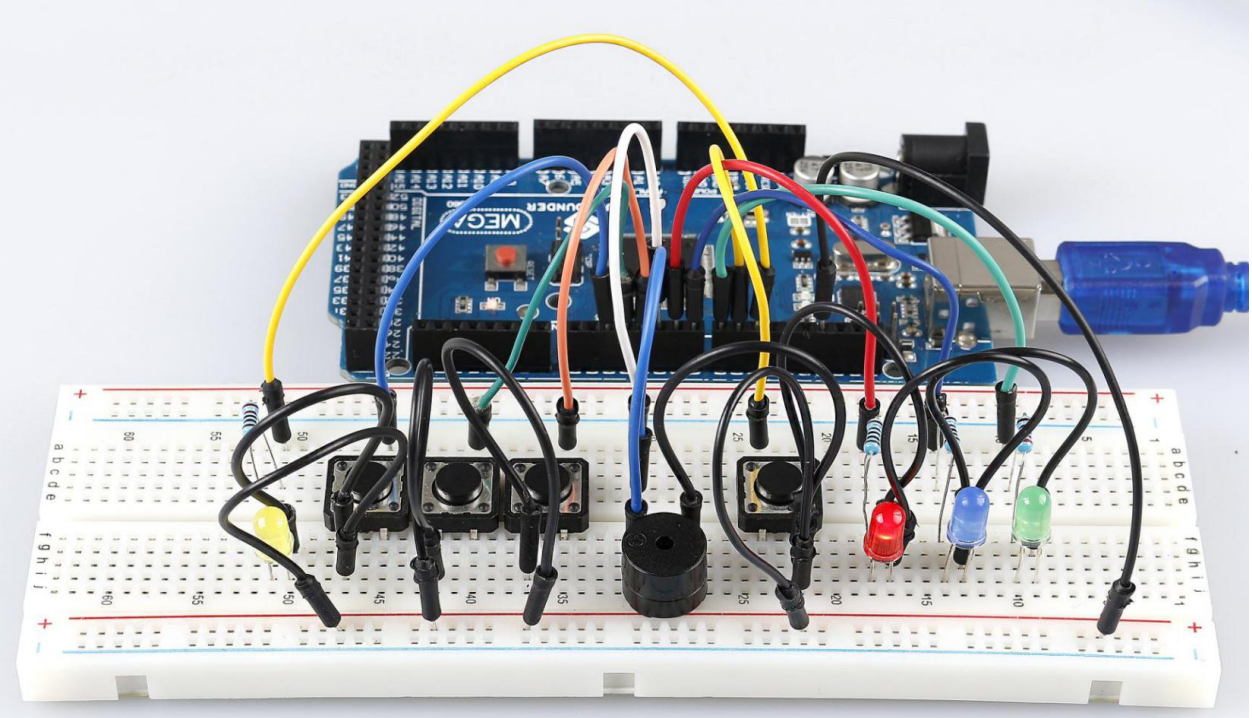


第 2 步：打开代码文件 `Lesson_21_Answer_Machine.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按键来上传代码。

现在，首先按下按键 4 开始。如果你先按下按键 1，你将看到相应的 LED 亮起，蜂鸣器将发出哔哔声。然后再次按下按键 4 进行重置，然后再按下其他按键。



7.21.5 代码

7.21.6 代码分析

这个实验的代码可能有点长。但是语法很简单。

这个代码用到了 6 个嵌套 if 语句。

- 第一个 if 语句用来判断按键 4 是否按下。
- 第二个 if 语句用来再次判断按键 4 是否按下，用来防止误触。若确定按下，则让 flag 为 1，同时让 LED 点亮。
- 第三个 if 语句用来判断 flag 的值，如果为 1(按键 4 已按下)，此时读取按键 1，2，3 的值。
- 第四-六个 if 语句用来分别判断按键 1，2，3 是否按键，如果按下，则让 LED 点亮，蜂鸣器出声音。

Alarm() 函数

```
void Alarm()
{
for(int i=0;i<100;i++){
    digitalWrite(buzzerPin,HIGH); //the buzzer sound
    delay(2);
    digitalWrite(buzzerPin,LOW);  //without sound
    delay(2);                     //when delay time changed,the frequency changed
}
}
```

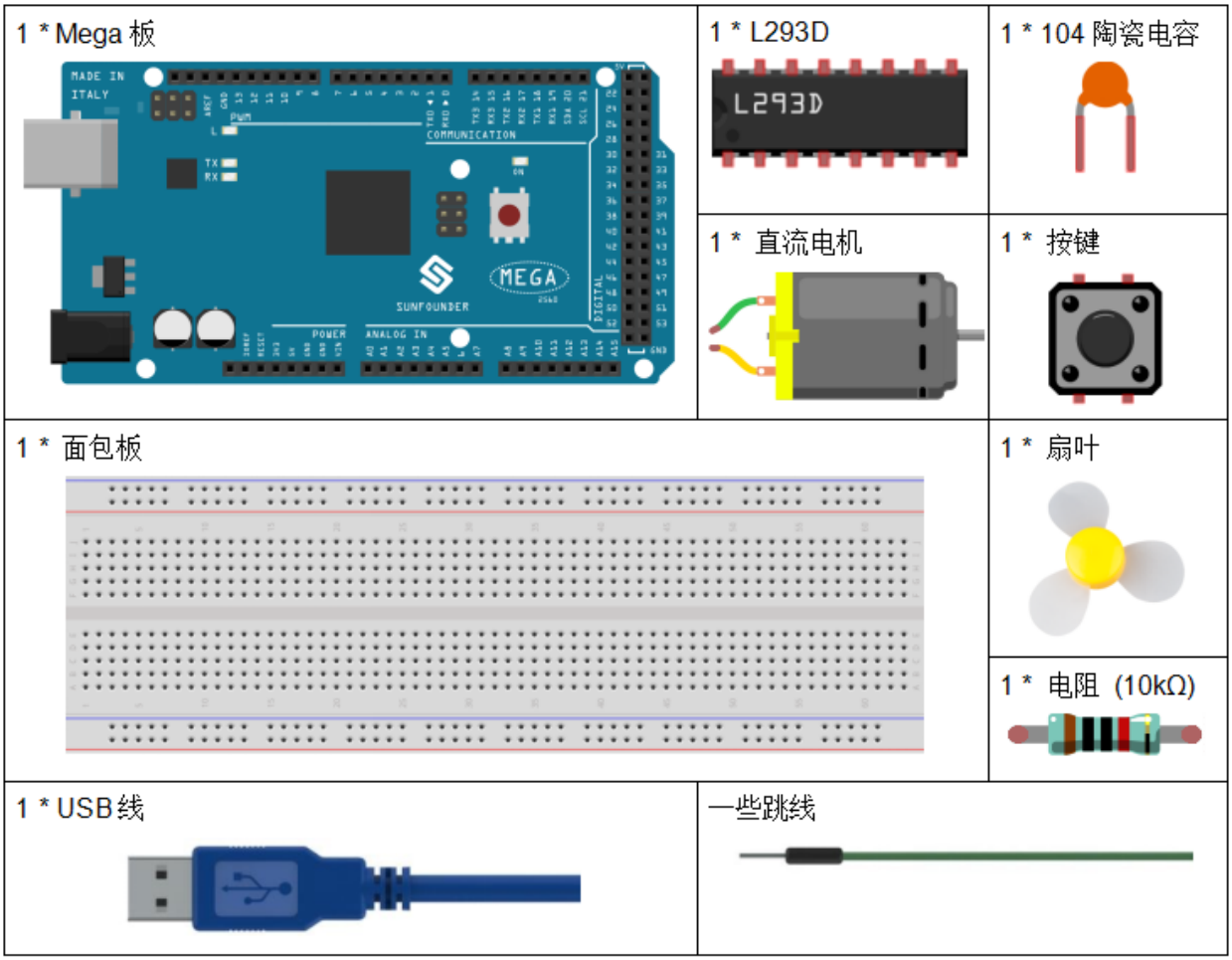
这个函数是用来设置蜂鸣器发出的声音长度和频率。

7.22 第 22 课简单创作 - 小风扇

7.22.1 介绍

在夏天若有个手持小风扇，让你能够走到哪吹到哪，并且还能更改档速，将是个非常爽的事情。
在这个课程中，我们将制作一个手持风扇原型 (去外壳)。

7.22.2 所需器件



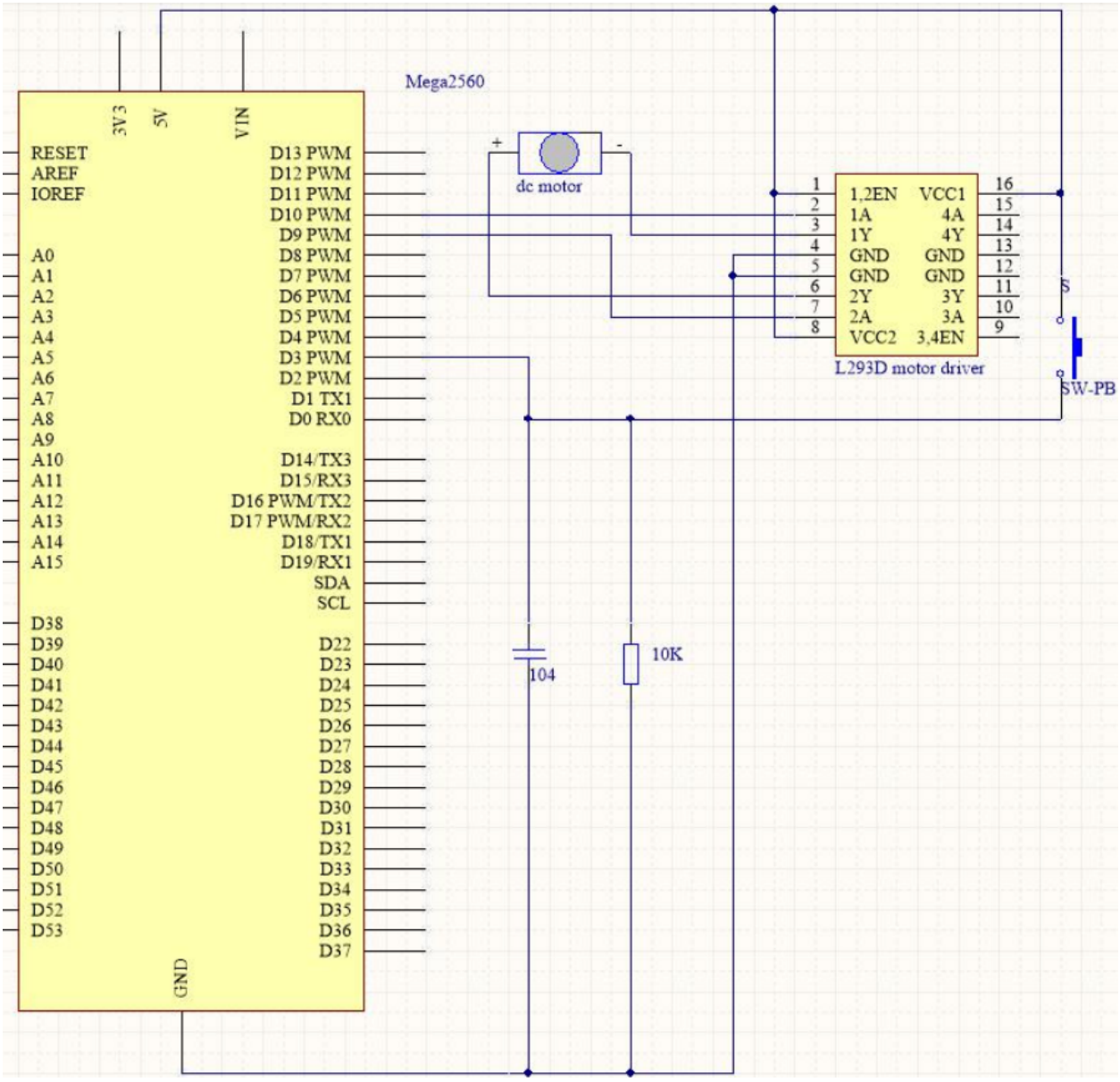
- *SunFounder Mega 板*
- 面包板
- 跳线
- 电阻
- 电容
- 按键
- *L293D*
- 直流电机

7.22.3 原理图

Arduino I/O 端口的最大电流为 20mA，但电机的驱动电流至少为 70mA。因此，我们不能直接使用 I/O 口来驱动电流；相反，我们可以使用 L293D 来驱动电机。

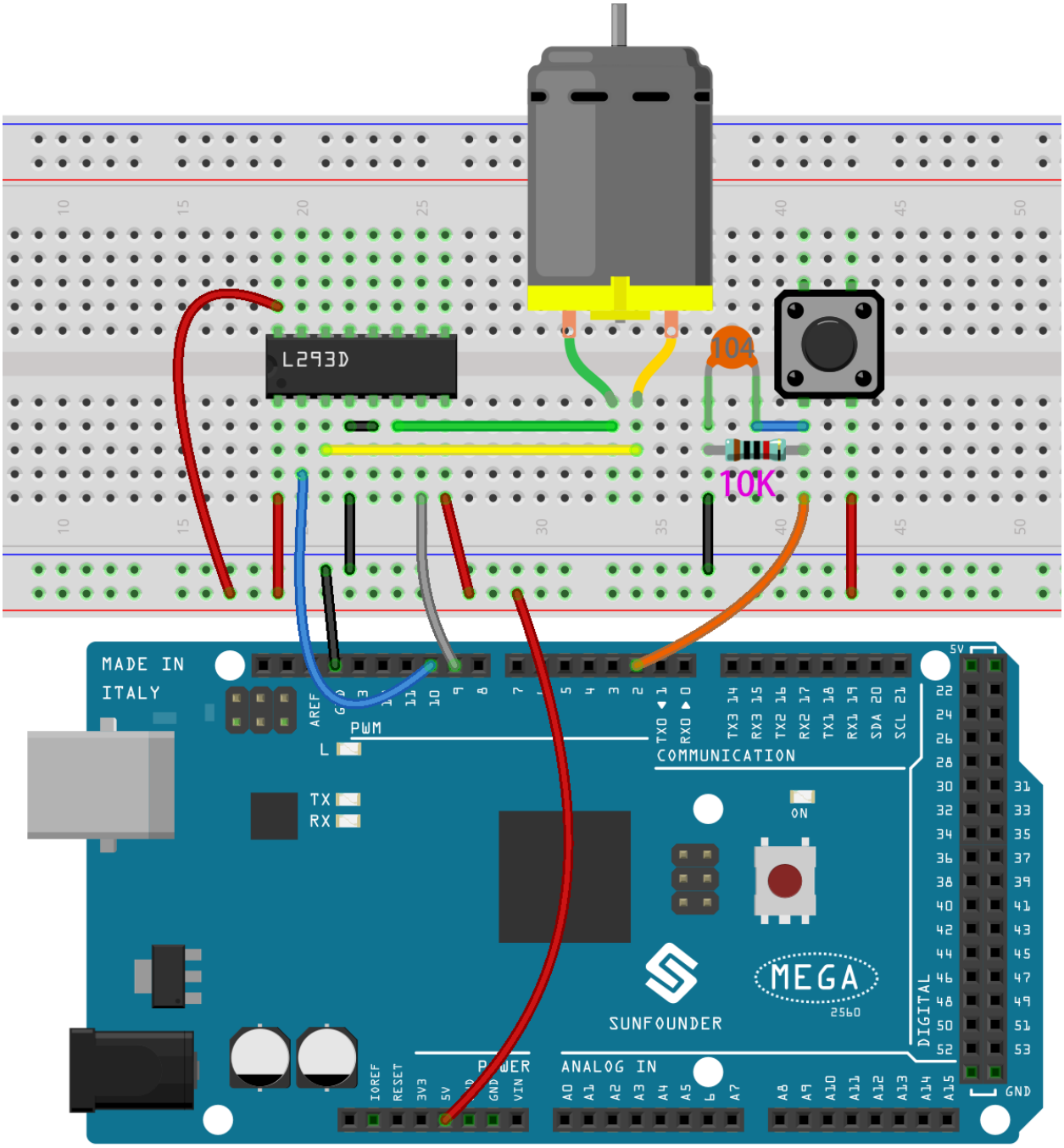
L293D 的 Enable pin 1,2EN 已经连接到 5V，所以 L293D 一直处于工作状态。将引脚 1A 和 2A 分别连接到控制板的引脚 9 和 10。电机的两个引脚分别连接到引脚 1Y 和 2Y。当 10 脚为高电平，9 脚为低电平时，电机开始向一个方向旋转。当引脚 10 为低电平且引脚 9 为高电平时，它以相反的方向旋转。

原理图如下所示：



7.22.4 实验步骤

第 1 步：搭建电路。

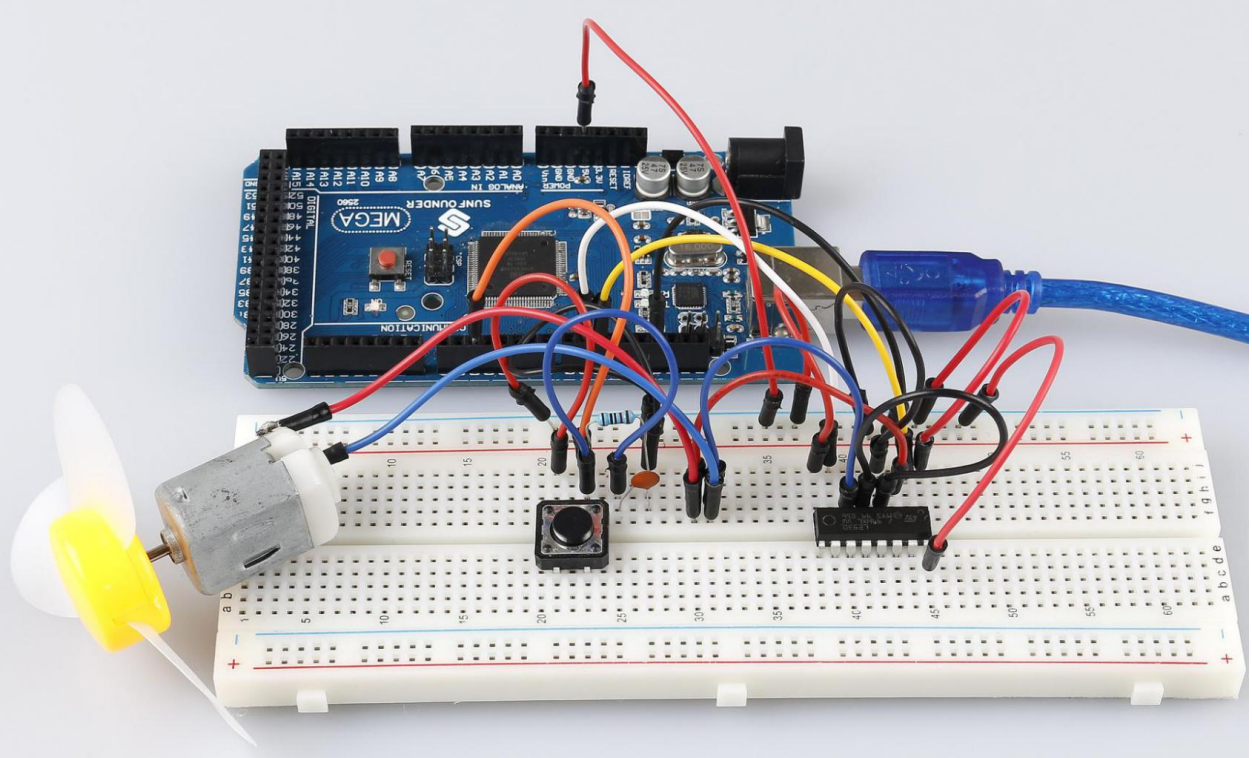


第 2 步：打开代码文件 `Lesson_22_Small_Fan.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

制作的小风扇有 3 档速度，按下第一次，风扇转速较慢，再次按下，风扇加速；第三次按下，风扇达到最大速度，按下第四次风扇将停止；以此为一个循环。



7.22.5 代码

7.22.6 代码分析

这个代码嵌套了 5 个 `if` 语句用来判断按键的按下状态。

- 第一个 if 语句用来判断按键是否按下。
- 第二个 if 语句用来判断时间是否过了 50ms。
- 第三个 if 语句用来判断过了 50ms, 按键确实有按下，以免有误触。
- 第四个 if 语句用来记录按键按下次数，每按下一次，`stat` 加 1。
- 第五个 if 语句用来判断按键按下次数是否大于 4，如果是，则将 `stat` 清零。

switch() 语句

```
switch(stat)
{
case 1:
    clockwise(rank1);// When stat=1, set the rotate speed of the motor as rank1=150
    break;
case 2:
    clockwise(rank2);// When stat=2, set the rotate speed of the motor as rank1=200
    break;
case 3:
    clockwise(rank3);// When stat=3, set the rotate speed of the motor as rank1=250
    break;
default:
    clockwise(0);
}
```

`switch` 语句与 `if` 语句一样，`switch case` 允许程序员在各种条件下执行的不同代码来控制程序流程。特别是，switch 语句将变量的值与 case 语句中指定的值进行比较。当找到值与变量的值匹配的 case 语句时，将运行该 case 语句中的代码。如果没有 `break` 语句，`switch` 语句将继续执行下面的表达式，直到 `break` 或到达 switch 语句的末尾。

在这部分代码中：

- 如果 stat = 1, 让风扇以速度 rank1(150) 转动。
- 如果 stat = 1, 让风扇以速度 rank2(200) 转动。
- 如果 stat = 1, 让风扇以速度 rank3(250) 转动。
- 如果 stat = 0, 让风扇以速度 0 转动。

clockwise() 函数

```
void clockwise(int Speed)//
{
    analogWrite(motorIn1,0);
    analogWrite(motorIn2,Speed);
}
```

该功能是设置电机的转速：将 `Speed` 写入引脚 9，将 0 写入引脚 10。电机朝某个方向旋转，速度为 `Speed` 的值。

7.22.7 实验总结

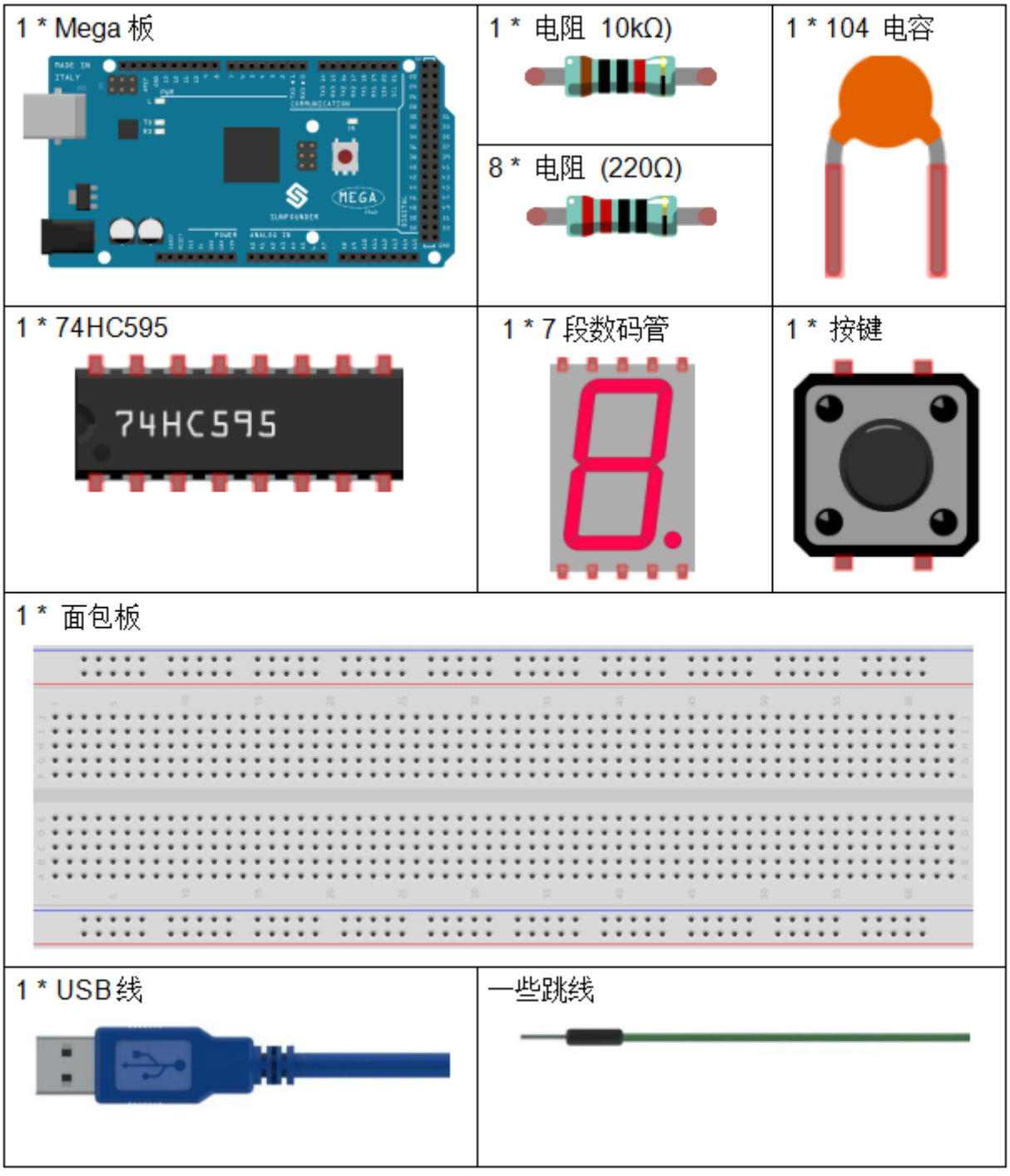
在本实验中，你还可以控制电机转动与否。只需将 L293D 的引脚 1、2EN 连接到控制板的 I/O 端口。设置 1、2EN 为高电平，电机开始转动；将其设置为低电平，它将停止旋转。

7.23 第 23 课简单创作 - 数字骰子

7.23.1 介绍

在之前的实验中，我们学习了如何使用 7 段数码管并通过按键控制 LED。在本课中，我们将使用一个 7 段数码管和一个按键来创建一个简单的数字骰子。

7.23.2 所需器件



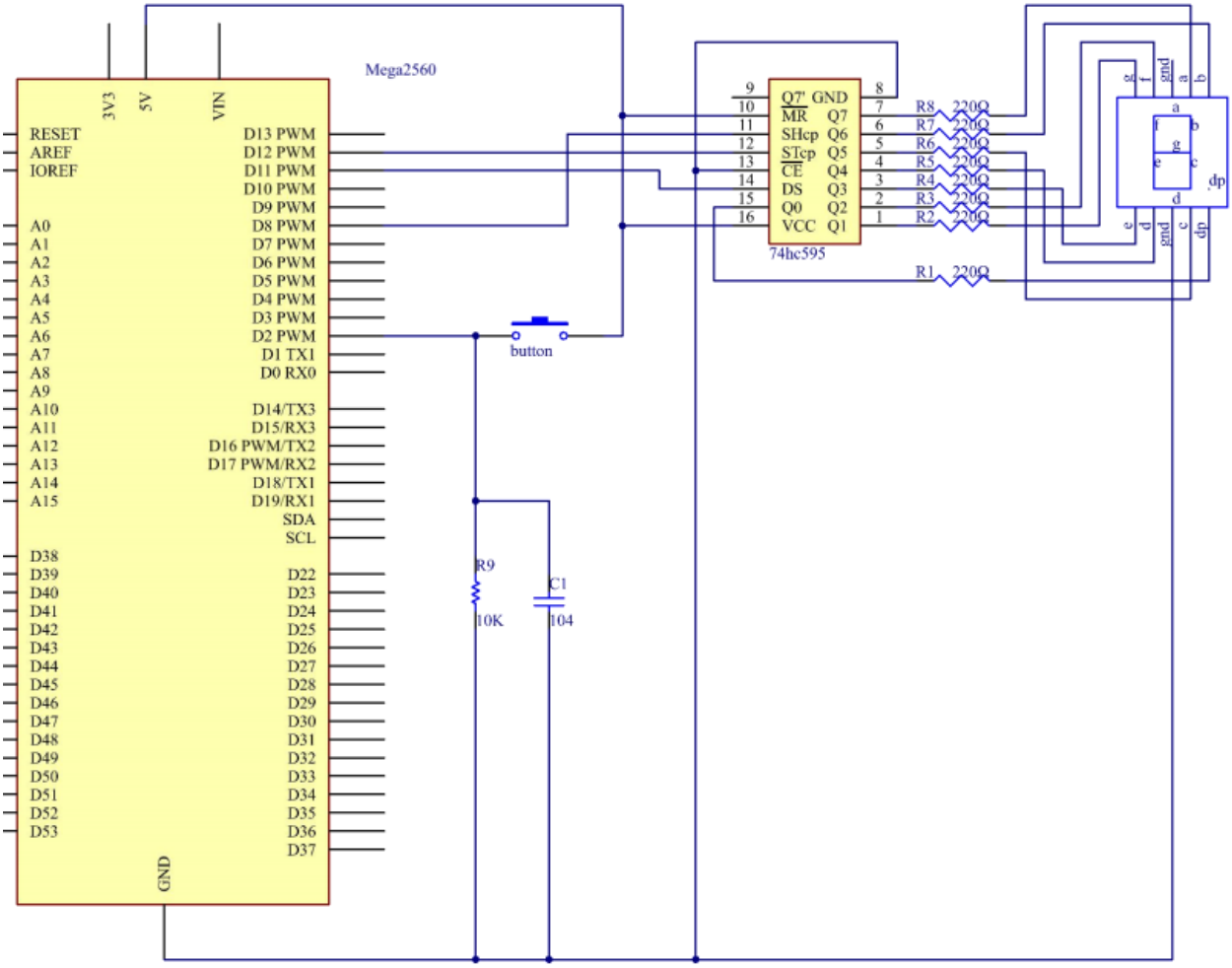
- *SunFounder Mega 板*
- *面包板*
- *跳线*
- *电阻*
- *7 段数码管*

- *74HC595*
- 按键
- 电容

7.23.3 原理图

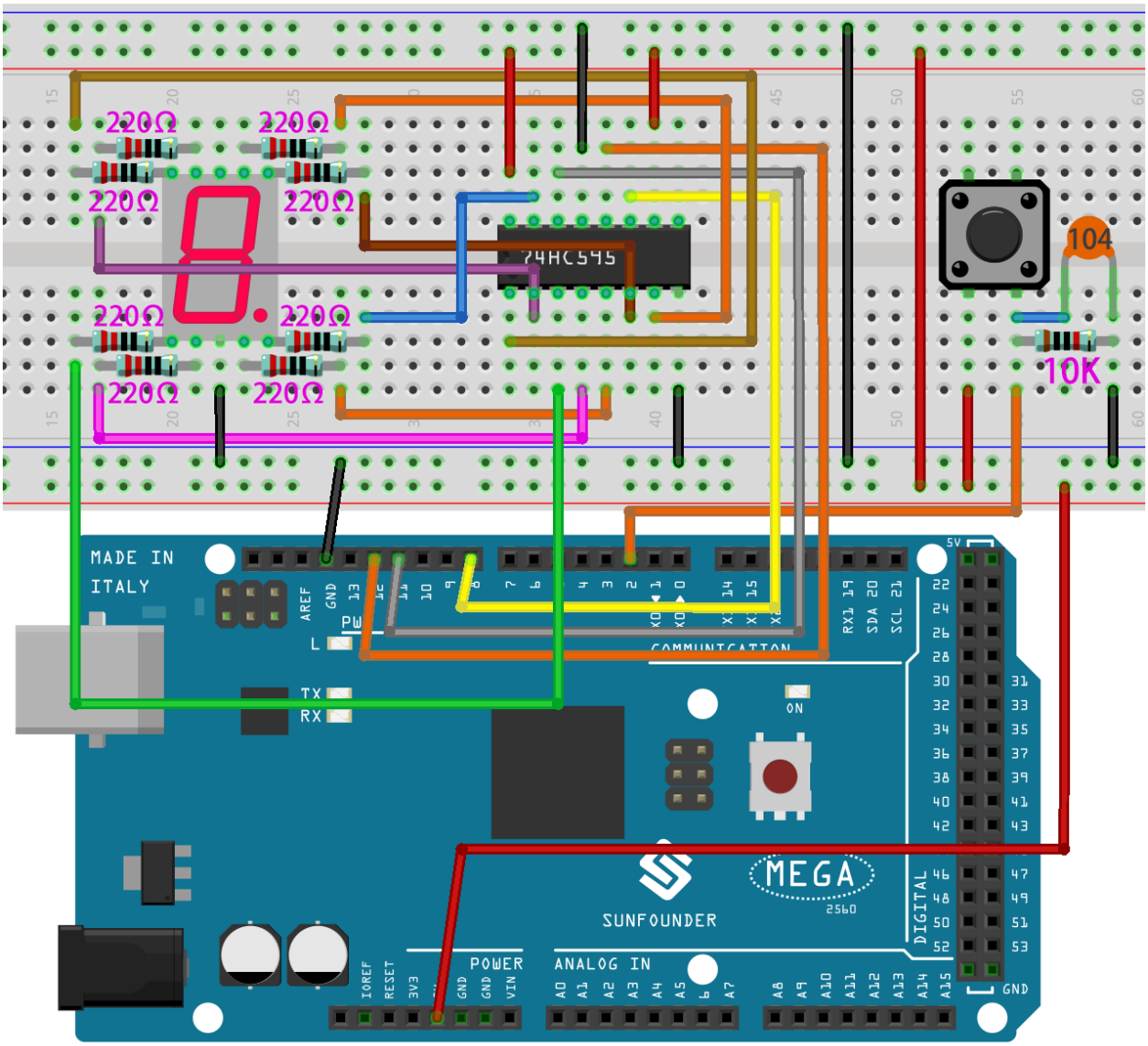
数字骰子背后的想法非常简单：一个 7 段数码管从 1 到 7 快速循环显示。当按下按键时，流动会减慢，直到它停在一个数字上。当再次按下按键时，该过程将重复。

原理图如下所示：



7.23.4 实验步骤

第 1 步：搭建电路。

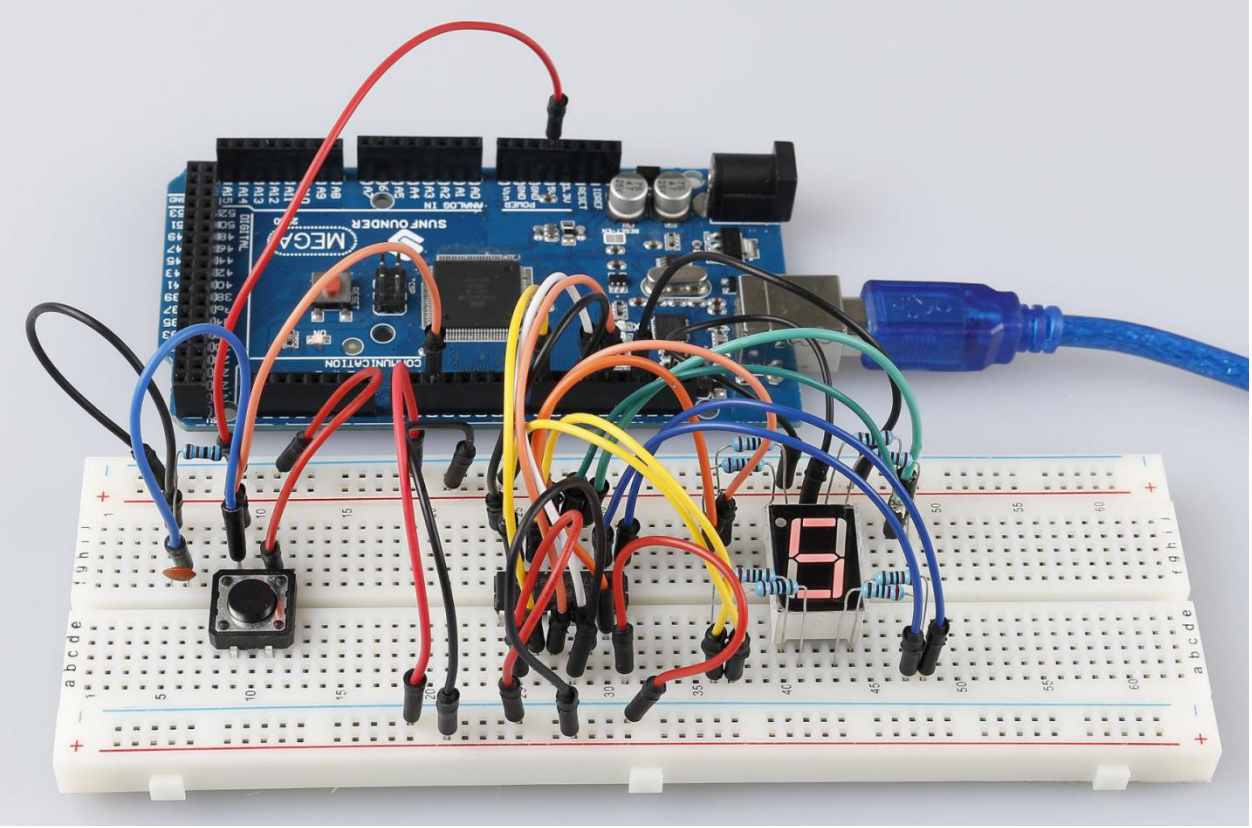


第 2 步：打开代码文件 `Lesson_23_Digital_Dice.ino`。

第 3 步：选择 **开发板**和 **端口**。

第 4 步：点击 **上传**按钮来上传代码。

你现在可以看到 7 段数码管循环显示 1~6。按下按键，显示速度会减慢，直到三秒后停止。再次按下按键，该过程将重复。



7.23.5 代码

7.23.6 代码分析

初始随机数来自 A0

```
randomSeed(analogRead(0));
```

初始随机数是从 A0 生成的，随机数的范围是 0-1023。

数字骰子

```
void loop()
{
    int stat = digitalRead(keyIn);  //store value read from keyIn
    if(stat == HIGH)  // check if the pushbutton is pressed
```

如果是，相应的引脚为高电平。

```
{
    num ++; // num adds 1
    if(num > 1)
    {
        num = 0;
    }
}
```

如果 num > 1，则清除该值。这是为了防止重复按压。所以不管你按多少次都算一次。

```
Serial.println(num);  // print the num on serial monitor
if(num == 1)  //when pushbutton is pressed
{
    randNumber = random(1,7); //Generate a random number in 1-7
    showNum(randNumber);  //show the randNumber on 7-segment
    delay(1000);  //wait for 1 second
    while(!digitalRead(keyIn));  //When not press button,program stop here.
```

让它一直显示最后一个随机数。

```
int stat = digitalRead(keyIn);
```

再次读取按键的状态。

```
if(stat == HIGH) // check if the pushbutton is pressed
```

如果是，请运行下面的代码。

```
{
        num ++; // num+1=2
        digitalWrite(ledPin,HIGH); //turn on the led
        delay(100);
        digitalWrite(ledPin,LOW); //turn off the led
        delay(100);
        if(num >= 1) // clear the num
        {
            num = 0;
        }
    }
    }
    //show random numbers at 100 microseconds intervals
    //If the button has not been pressed
    randNumber = random(1,7);
    showNum(randNumber);
    delay(100);
}
```

showNum() 函数

```
void showNum(int num)
{
    digitalWrite(latchPin,LOW); //ground latchPin and hold low for transmitting
    shiftOut(dataPin,clockPin,MSBFIRST,datArray[num]);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(latchPin,HIGH); //pull the latchPin to save the data
}
```

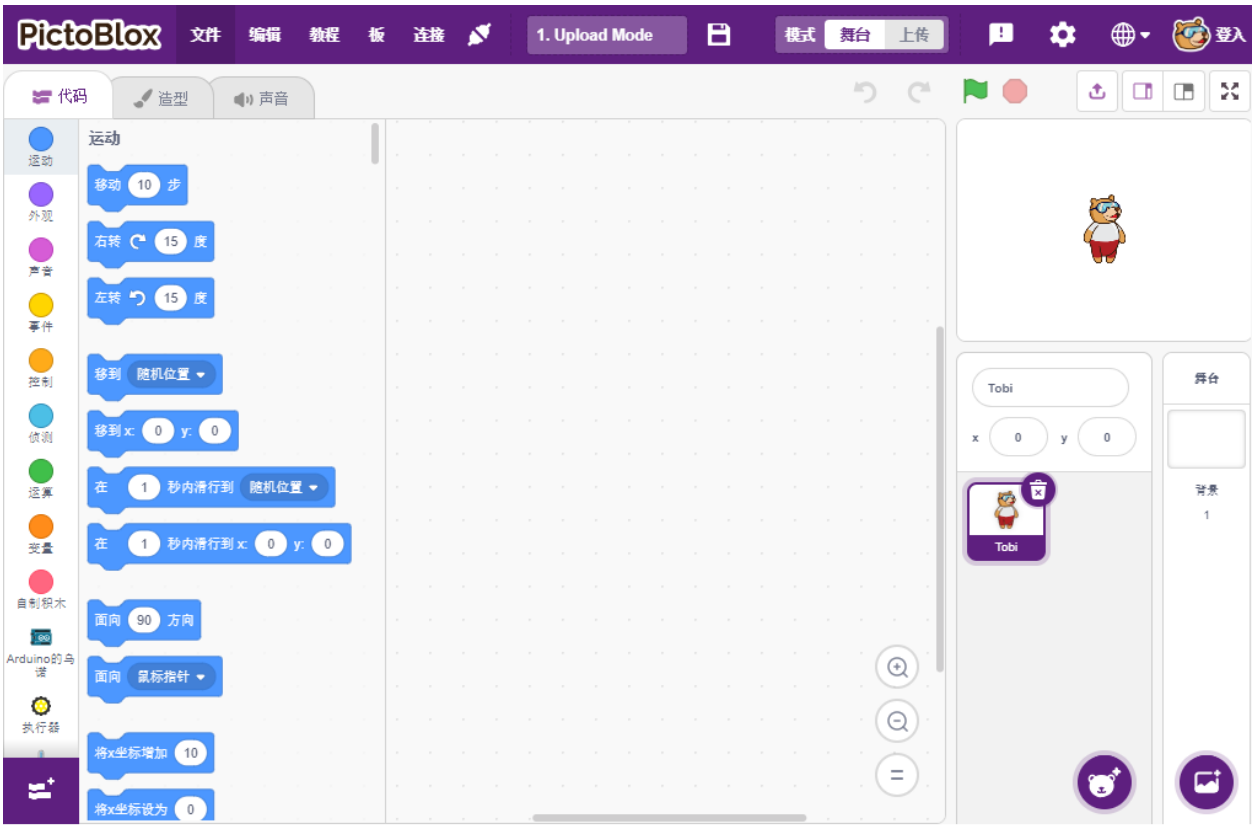
该功能是在 7 段数码管上显示 `dataArray[]` 中的数字。

CHAPTER 8

Scratch 项目

除了用 Uno/Mega2560 板在 Arduino IDE 上进行编程，我们还可以用它们来进行图形化编程。

这里我们推荐使用 Scratch 进行编程，但官方的 Scratch 目前只兼容树莓派，所以我们合作了一家公司, STEMPedia, 他们研发了一款适用于 Arduino 板子（Uno，Mega2560 和 Nano）的基于 Scratch 3 的图形化编程软件- PictoBlox.



它保留了 Scratch 3 的基本功能，还加入了其他的主控板，比如 Arduino Uno, Mega，Nano，ESP32，Microbit 及 STEAMPedia 自制的主控板，可以用外接的传感器，机器人来控制舞台上的精灵，具有很强的硬件交互功能。

除此之外，它还有 AI 和机器学习，即使你没有太多的编程基础，你也可以学习和使用这些流行和高新技术。

只需拖放 Scratch 编码块，即可制作酷炫的游戏、动画、互动项目，甚至以你想要的方式控制机器人！

现在让我们开始探索旅程吧！

8.1 安装 PictoBlox

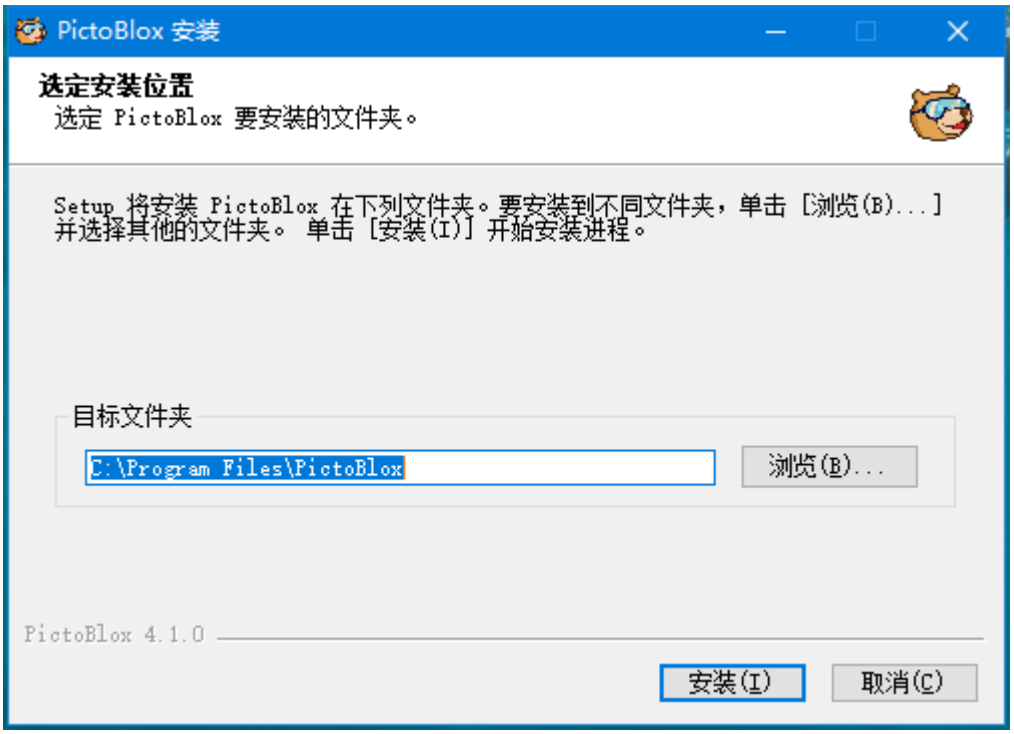
根据你的系统来选择安装指示：

- *Windows*
- *macOS*

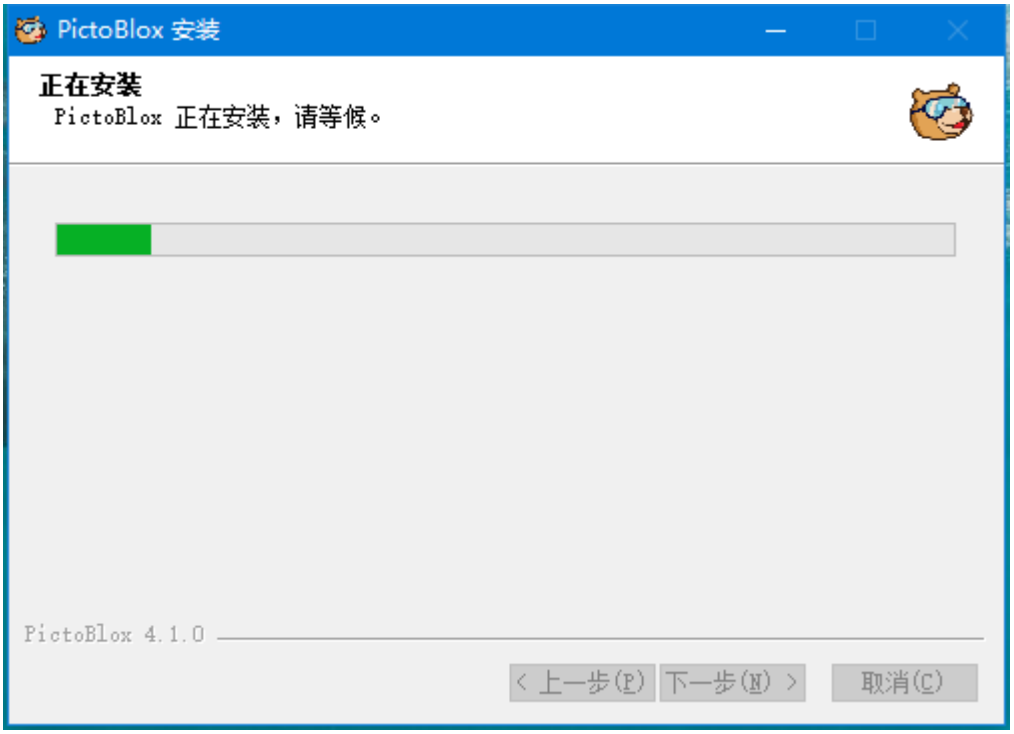
8.1.1 Windows

参考章节下载资料 来下载相关的资料，然后进入到 `SunFounder Uno R3 学习套件\编程软件\PictoBlox\Windows` 路径中，在这个文件夹中包含了 32 位和 64 位的安装包，请根据你的系统来选择。

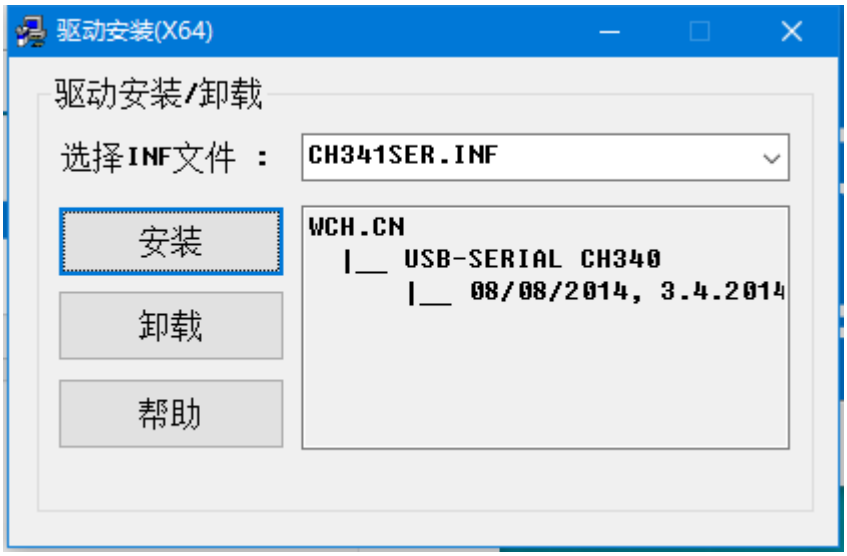
双击下载的.exe 文件，将会出现让你选择安装路径的窗口，然后点击 **安装**。



等待安装进行。

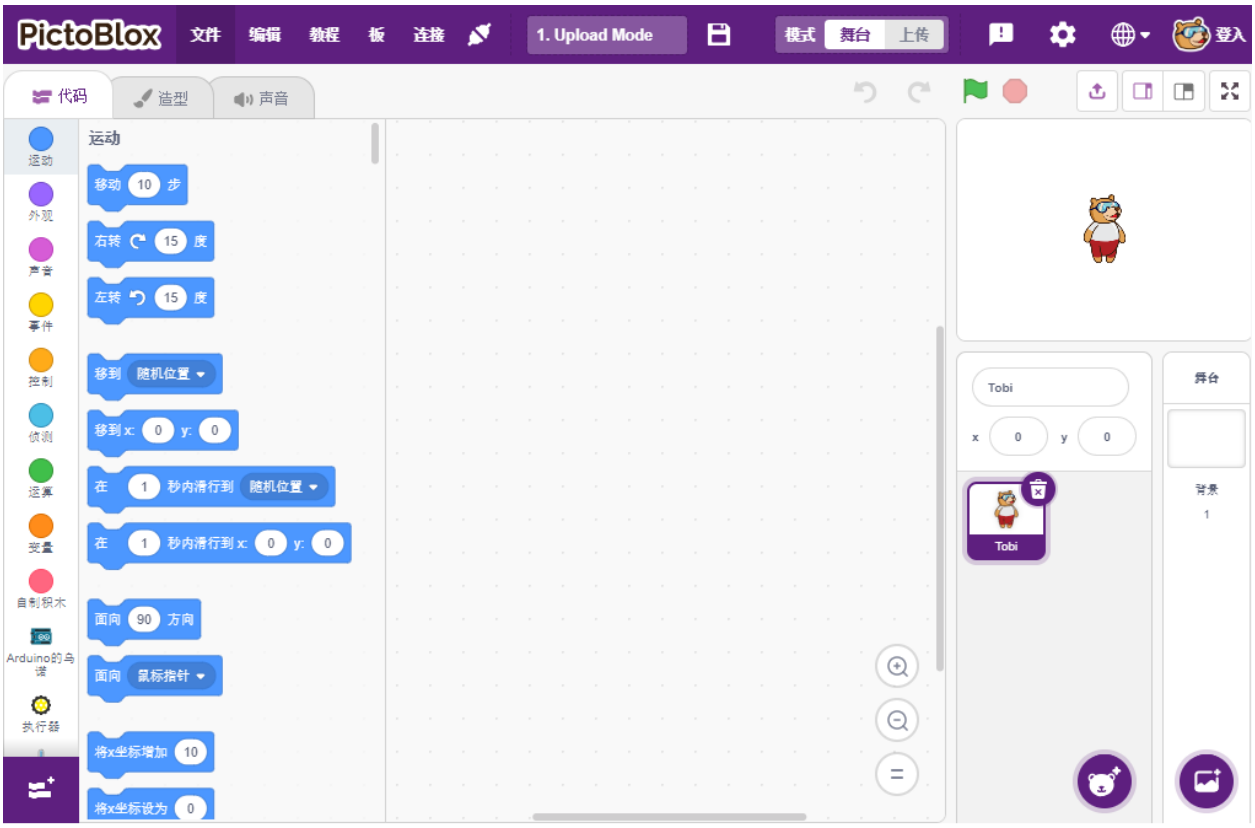


安装相应的驱动，过一段时间后将会提示”驱动预安装成功“。



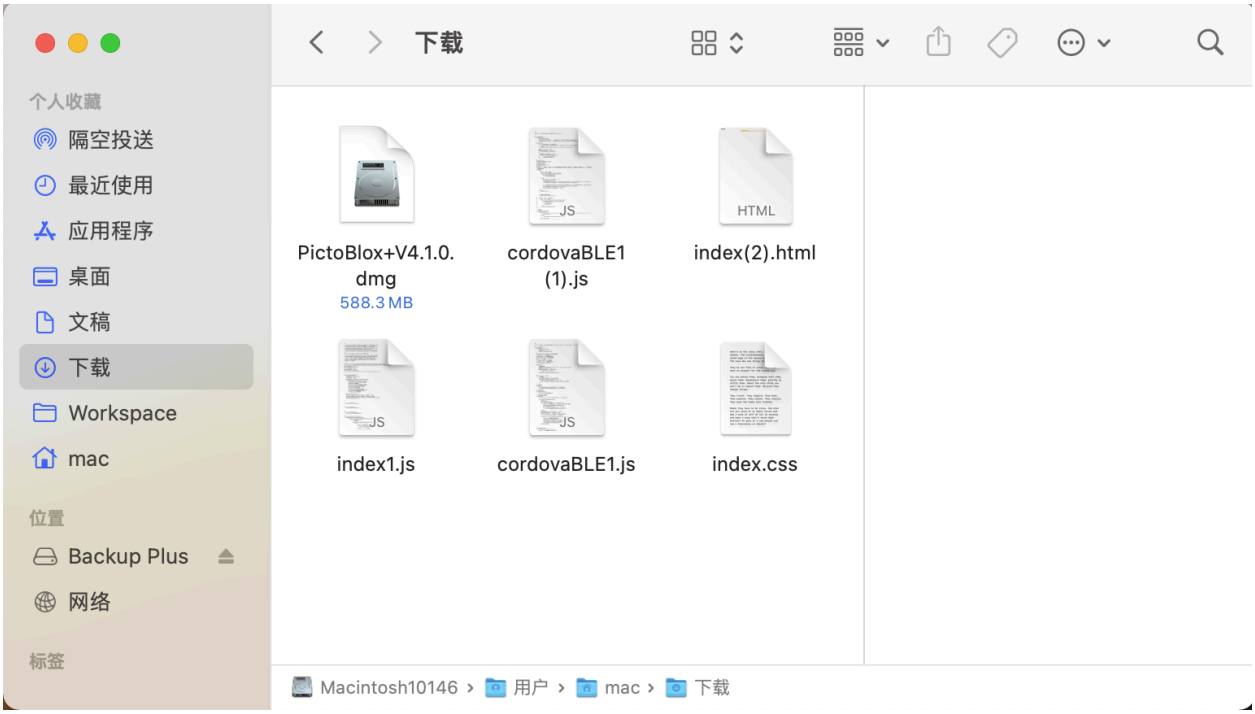
点击 **完成**来关闭安装向导。

此时 PictoBlox 将被打开，你可以根据它的使用向导来简单了解下 PictoBlox 的使用。

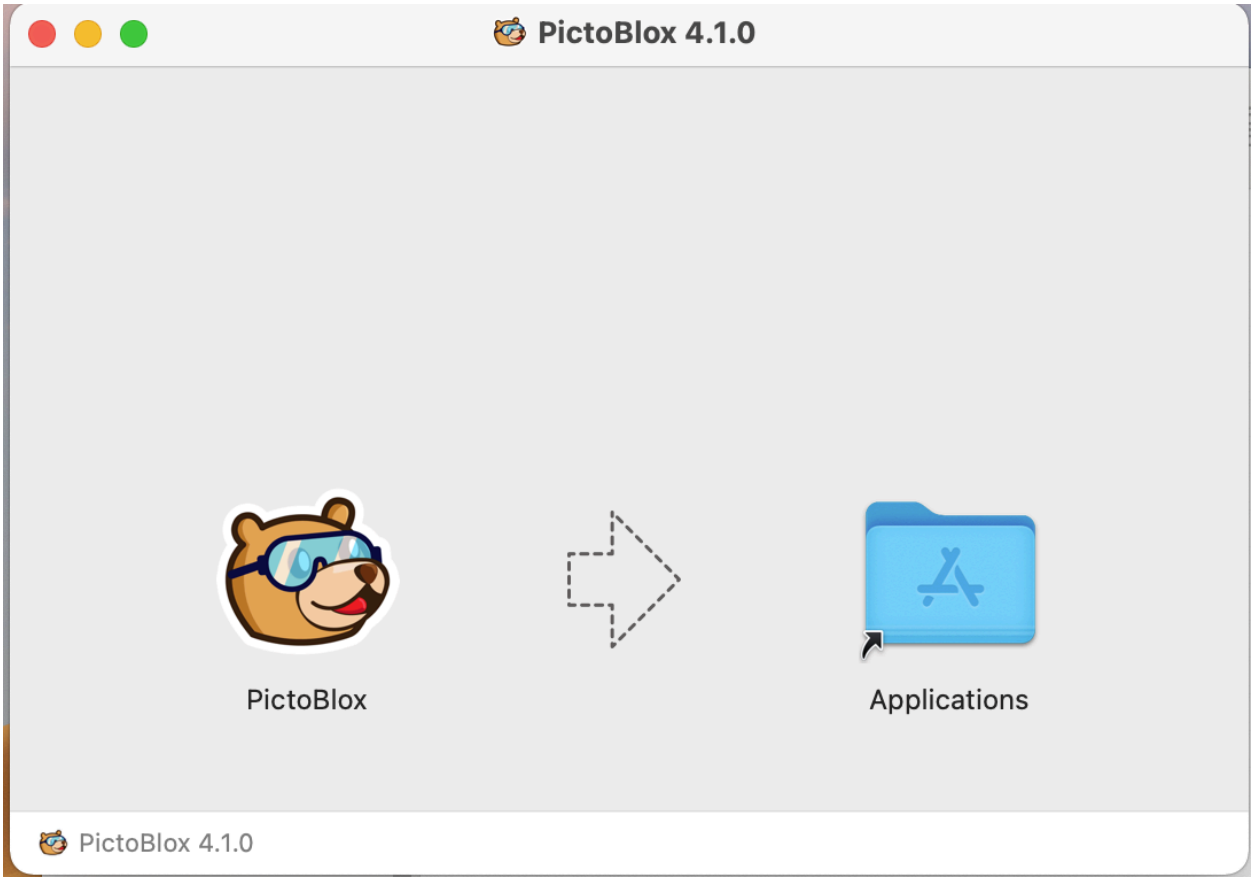


8.1.2 macOS

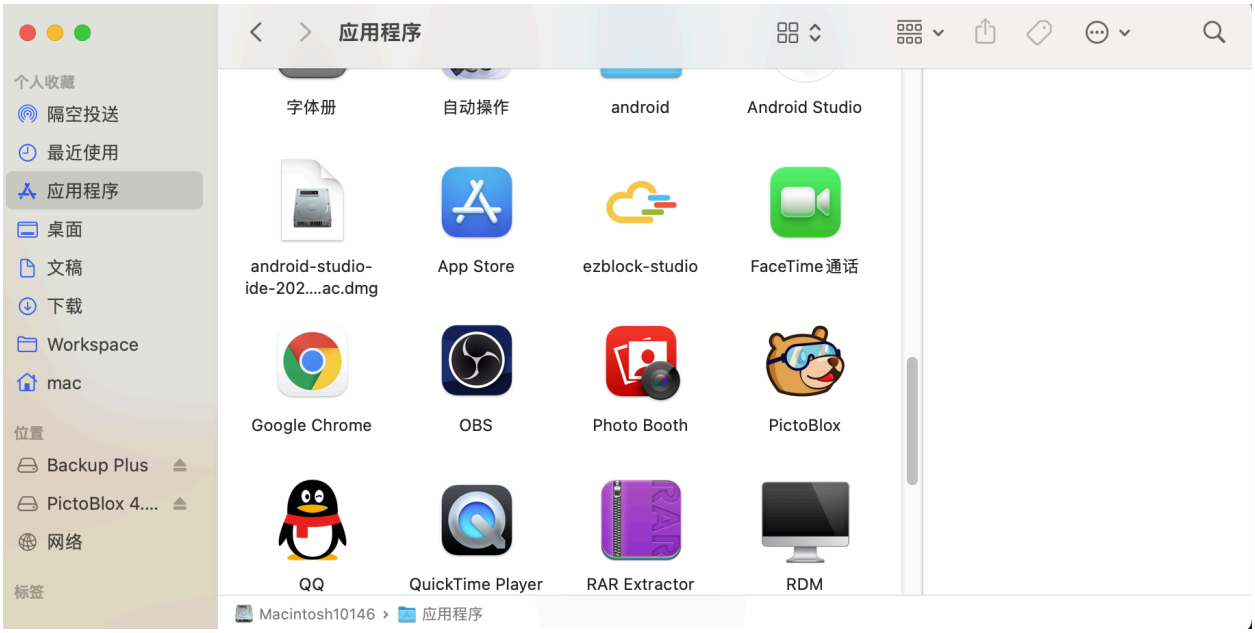
参考章节下载资料 来下载相关的资料，然后进入到 SunFounder Uno R3 学习套件\编程软件\PictoBlox\macOS 路径中，双击 .dmg 文件。



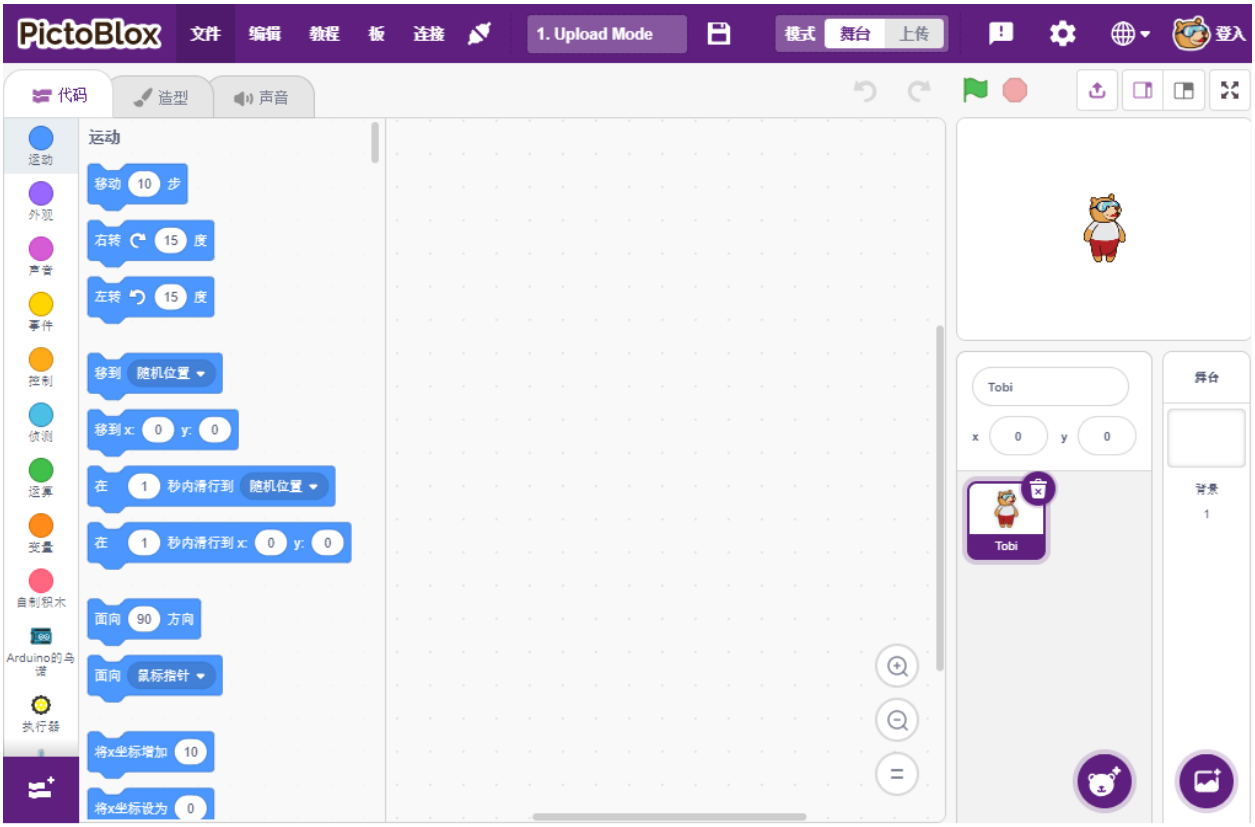
将它拖到 **Application（应用程序）**文件夹。



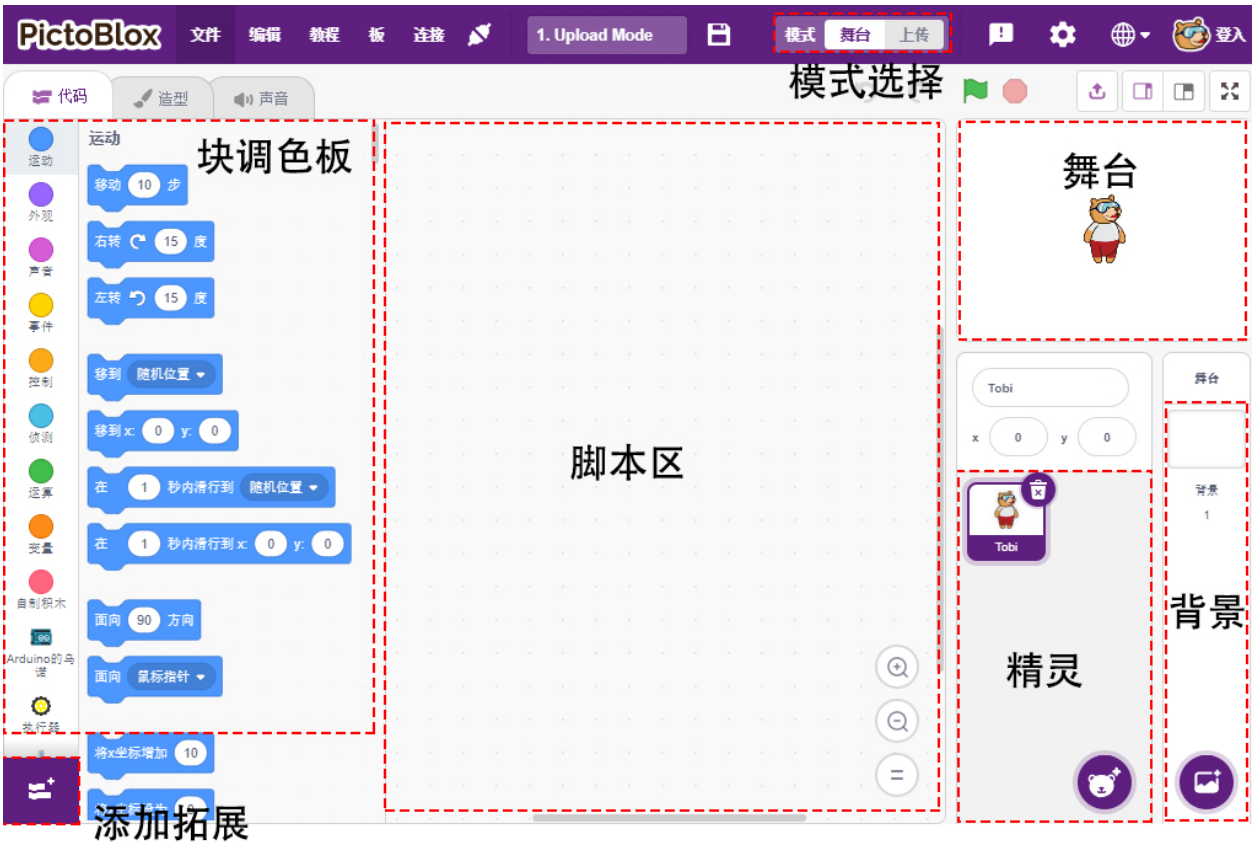
现在你将在应用程序文件夹看到 PictoBlox。



双击来将它打开，会有提示想要访问摄像头和麦克风，看你自己选择是否允许。打开 PictoBlox 后，你可以根据它的使用向导来简单了解下 PictoBlox 的使用。


PictoBlox
文件
编辑
教程
板
连接
1. Upload Mode
模式
舞台
上传
登入
代码
造型
声音
运动
外观
声音
事件
控制
侦测
运算
变量
自制积木
Arduino的乌诺
执行器
移动 10 步
右转 15 度
左转 15 度
移到 随机位置
移到 x: 0 y: 0
在 1 秒内滑行到 随机位置
在 1 秒内滑行到 x: 0 y: 0
面向 90 方向
面向 鼠标指针
将x坐标增加 10
将x坐标设为 0
Tobi
x 0 y 0
Tobi
舞台
背景
1

8.2 界面介绍



精灵

精灵是在项目中执行不同操作的对象或角色，它理解并服从给予它的命令。每个精灵都有特定的造型和声音，你也可以自定义。

舞台

舞台是精灵根据你的程序在背景中执行动作的区域。

背景

背景是用来装饰舞台的。你可以从 PictoBlox 中选择背景、自己画一个或从你的计算机上传图像。

脚本区

脚本是 PictoBlox/Scratch 术语中的程序或代码。它是一组按特定顺序排列的“块”，用于执行一个任务或一系列任务。你可以编写多个脚本，所有脚本都可以同时运行。你只能在屏幕中央的脚本区编写脚本。

块

块就像拼图的碎片，用于通过简单地将它们堆叠在脚本区域中来编写程序。使用块来编写代码可以使编程更容易并降低出错的概率。

块调色板

块调色板位于左侧区域，并以其功能命名，例如运动、声音和控制。每个调色板都有不同的块，例如，**运动**调色板中的块将控制精灵的移动，而 **控制**调色板中的块将根据特定条件控制脚本的工作。

还有其他类型的块调色板可以从位于左下角的添加扩展按钮加载。

模式

与 Scratch 不同，PictoBlox 有两种模式：

- 舞台模式: 在此模式下，你可以为精灵和板编写脚本以与精灵实时交互。如果你断开电路板与 Pictoblox 的连接，你将无法再进行交互。
- 上传模式: 此模式允许你编写脚本并将其上传到开发板，以便你在未连接计算机的情况下也可以使用，例如你需要上传脚本来制作移动机器人。

更多的信息请参考：https://thestempedia.com/tutorials/getting-started-pictoblox

8.3 项目

以下项目是按照编程难度的顺序编写的，因此我们建议你按顺序阅读它们。

在每个项目中，都有非常详细的步骤教你如何搭建电路，如何一步步编程，达到最终的效果。

当然，如果你想直接打开脚本运行它，你可以参考下载资料，我们已经把所有的代码都上传到了网盘。

下载完成后，解压。请参阅舞台模式 以直接运行单个脚本。

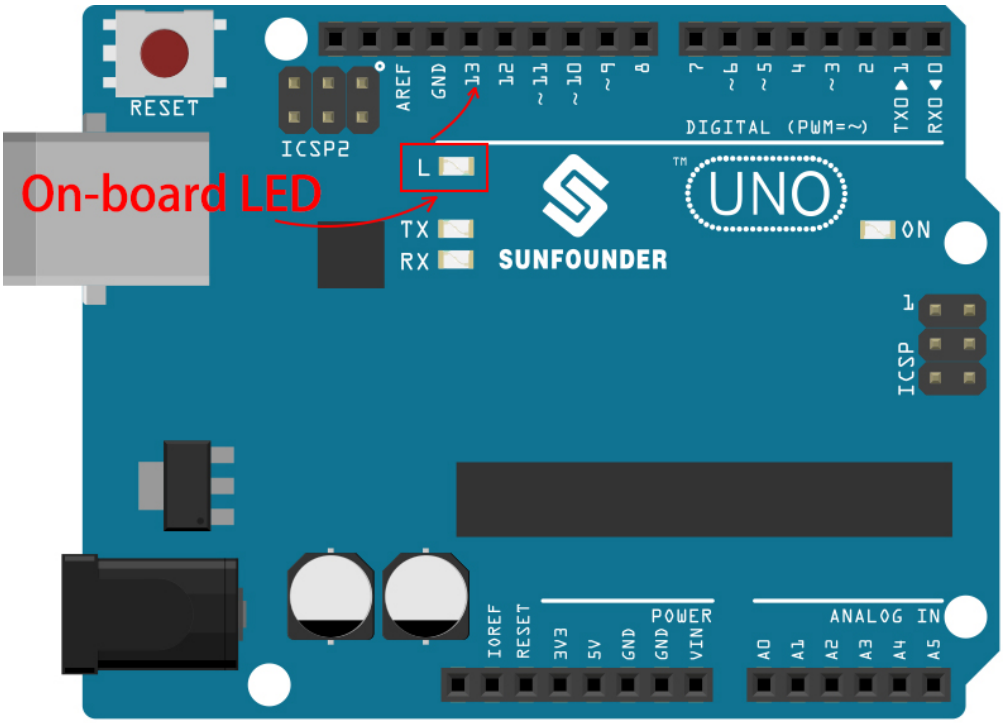
但是第 11 个项目是在 Upload 模式下读取温湿度值，请参考上传模式。

备注： 在示例中，使用了 Arduino Uno。如果你使用 Arduino Mega2560，电路搭建和编程步骤基本相同，唯一不同的是你需要在编程前为 Board 选择 Arduino Mega。

8.3.1 1. PictoBlox 的快速使用指南

现在开始制作我们的第一个项目来学习 PictoBlox 的 2 种模式。

另外，在 Arduino Uno/Mega2560 上有一个内嵌的 LED 连接在 Pin 13，我们将学习在 2 个不同的模式下，让这个 LED 闪烁。



你将学习

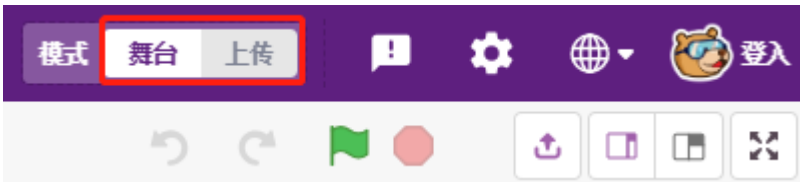
- 舞台和上传模式
- 给引脚设置高低电平
- 设置时间间隔和让脚本无限循环

舞台模式

1. 连接到 Arduino 板

用一根 USB 线将你的 Arduino 板子连接到电脑上，一般电脑会自动识别你的板子，最后分配一个 COM 端口.

打开 PictoBlox, 你会看到右上角可进行模式切换。默认是 **舞台模式**，Tobi 站在舞台上。



在右上角的导航栏点击 **板**。

比如选择 Arduino Uno

随即会弹出连接窗口让你选择特定的 **设备名称 COMxx** 来连接，连接完成后回到主页。若在使用过程中断链，你也可以重新连接。

于此同时，与 Arduino Uno 相关的调色板，比如，Arduino 的乌诺, 执行器等将出现在调色板区域中。

2. 上传固件

由于我们要在 **舞台模式**下工作，我们必须将固件上传到板上，它将确保电路板和计算机之间的实时通信，上传固件是一个一次性过程。为此，请单击 **上传固件**按钮。等待一段时间后，上传成功提示将会出现。

备注： 如果你是第一次在 PictoBlox 中使用这个 Arduino 板子，或者是这个 Arduino 之前用 Arduino IDE 上传的代码。那你在使用前，需要点 **上传固件**。

3. 编程

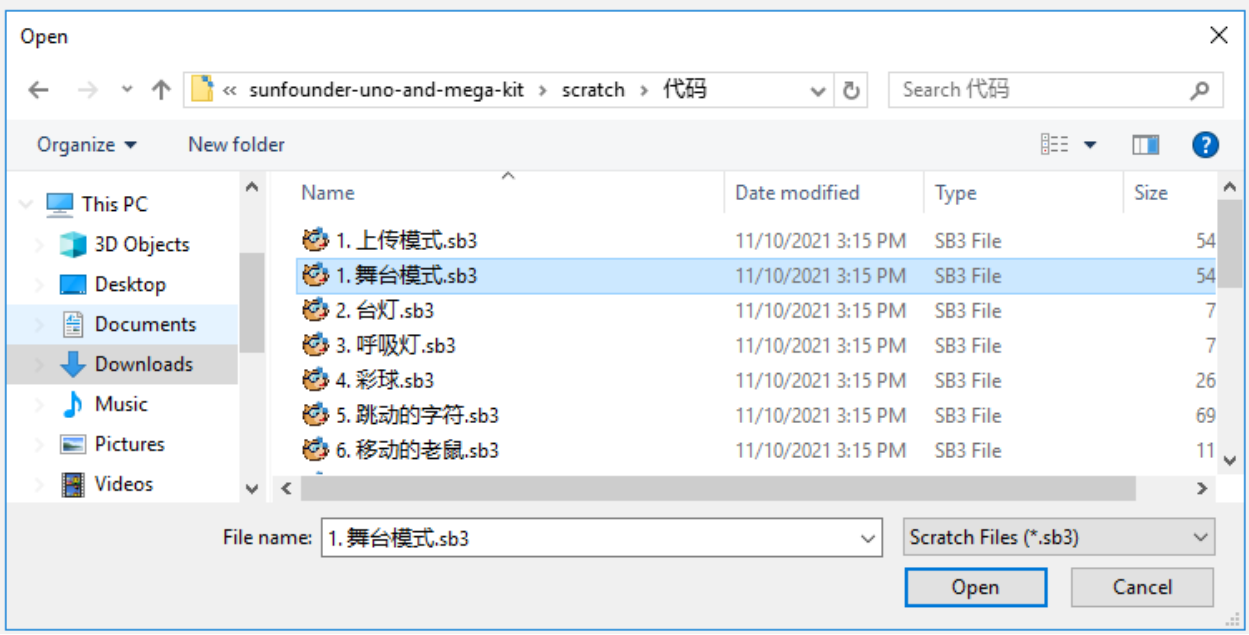
- 直接打开和运行脚本

当然，如果你想直接打开脚本运行它，你可以参考下载资料，我们已经将所有脚本都已经上传。

你可以单击右上角的 **文件 -》 打开**。

选择 **从计算机上打开**.

去到这个路径: SunFounder Uno R3 学习套件\Scratch 项目代码\代码来打开 **1. 舞台模式.sb3**。请确保你已经参考下载资料 来下载所需的代码.



直接点击脚本运行，有些项目是点击绿旗或者点击精灵。

- 一步一步编程

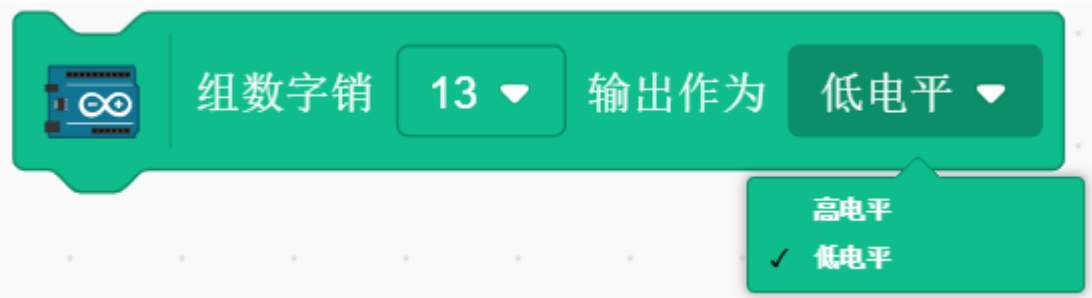
你还可以按照以下步骤逐步编写脚本。

单击 **Arduino 乌诺**调色板。

Arduino 板上的 LED 由数字引脚 13（只有 2 个状态高或低）控制，因此将 [组数字销 () 输出作为（高/低电平）] 块拖到脚本区域。

由于 LED 的默认状态是点亮的，现在将引脚 13 设置为低电平并单击此块，你将看到 LED 熄灭。

- [组数字销 () 输出作为（高/低电平）]：将数字引脚（2~13）设置为（高/低）电平。



为了看到 LED 连续闪烁的效果，需要使用 **控制**调色板中的 [等待 1 秒] 和 [重复执行] 块，如下图，写入后点击这些块，有黄色光晕意味着它正在运行。

- [等待 1 秒]：来自控制调色板，用于设置 2 个块之间的时间间隔。
- [重复执行]：来自控制调色板，允许脚本继续运行，除非手动暂停。

上传模式

1. 连接 Arduino 板

用一根 USB 线将你的 Arduino 板子连接到电脑上，一般电脑会自动识别你的板子，最后分配一个 COM 端口.

在右上角的导航栏点击 **板**。

例如，选择 **Arduino Uno**.

随即会弹出连接窗口让你选择特定的 **设备名称 COMxx** 来连接，连接完成后回到主页。若在使用过程中断链，你也可以重新连接。

于此同时，与 Arduino Uno 相关的调色板，比如，Arduino 的乌诺, 执行器等将出现在调色板区域中。选择上传模式后，舞台区域就会替换成 Arduino 源码。

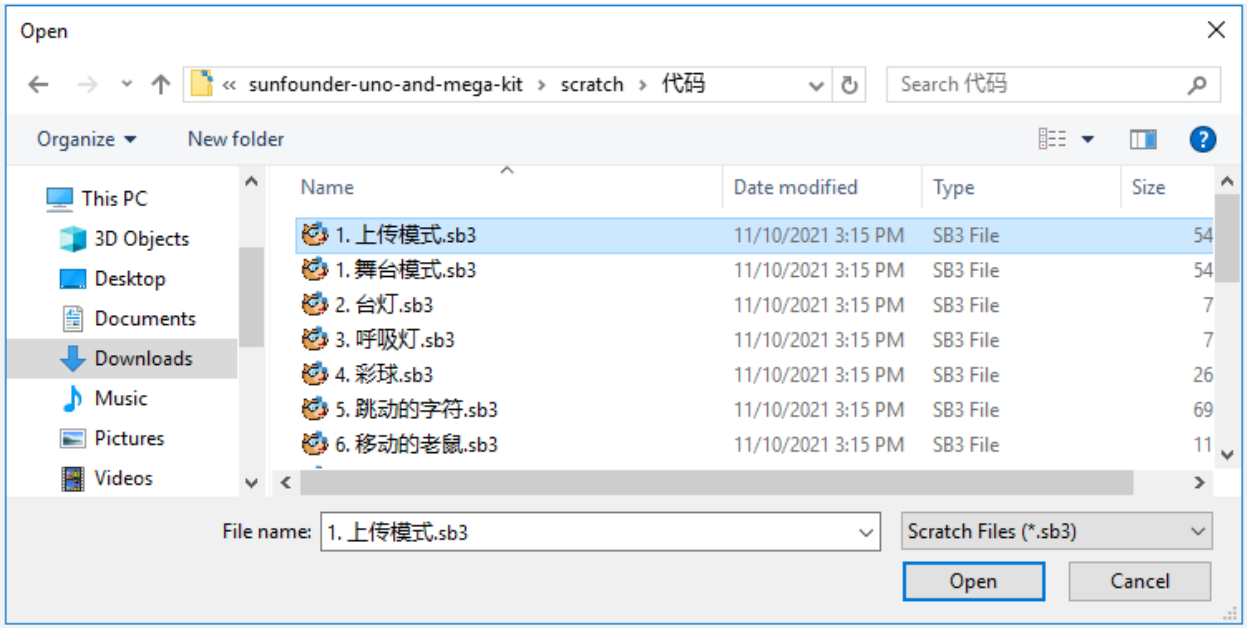
2. 编程

- 直接打开和运行脚本

你可以单击右上角的 **文件 -》 打开**。

选择 **从计算机上打开**.

去到这个路径: `SunFounder Uno R3 学习套件\Scratch 项目代码\代码`来打开 **1. 上传模式.sb3**。请确保你已经参考下载资料 来下载所需的代码.



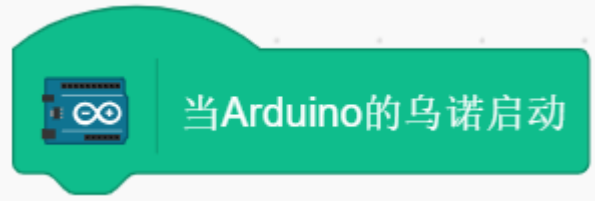
最后点击 **上传代码**按钮。

- 一步步编程

你还可以按照以下步骤逐步编写脚本。

单击 **Arduino 乌诺**调色板。

将 [Arduino 的乌诺启动] 块拖到脚本区域，这个在上传模式中每个脚本都需要的。



Arduino 板上的 LED 由数字引脚 13（只有 2 个状态高或低）控制，因此将 [组数字销 () 输出作为（高/低电平] 块拖到脚本区域。

由于 LED 的默认状态是点亮的，现在将引脚 13 设置为低电平并单击此块，你将看到 LED 熄灭。

- [组数字销 () 输出作为（高/低电平）]：将数字引脚（2~13）设置为（高/低）电平。

此时你将看到 Arduino 代码显示在右侧，如果你想要编辑这个代码，你可以将 **编辑模式**打开。

为了看到 LED 连续闪烁的效果，需要使用 **控制**调色板中的 [等待 1 秒] 和 [重复执行] 块，如下图，写入后点击这些块，有黄色光晕意味着它正在运行。

- [等待 1 秒]：来自控制调色板，用于设置 2 个块之间的时间间隔。
- [重复执行]：来自控制调色板，允许脚本继续运行，除非手动暂停。


当Arduino的乌诺启动
重复执行
组数字销 13 输出作为 低电平
等待 1 秒
组数字销 13 输出作为 高电平
等待 1 秒

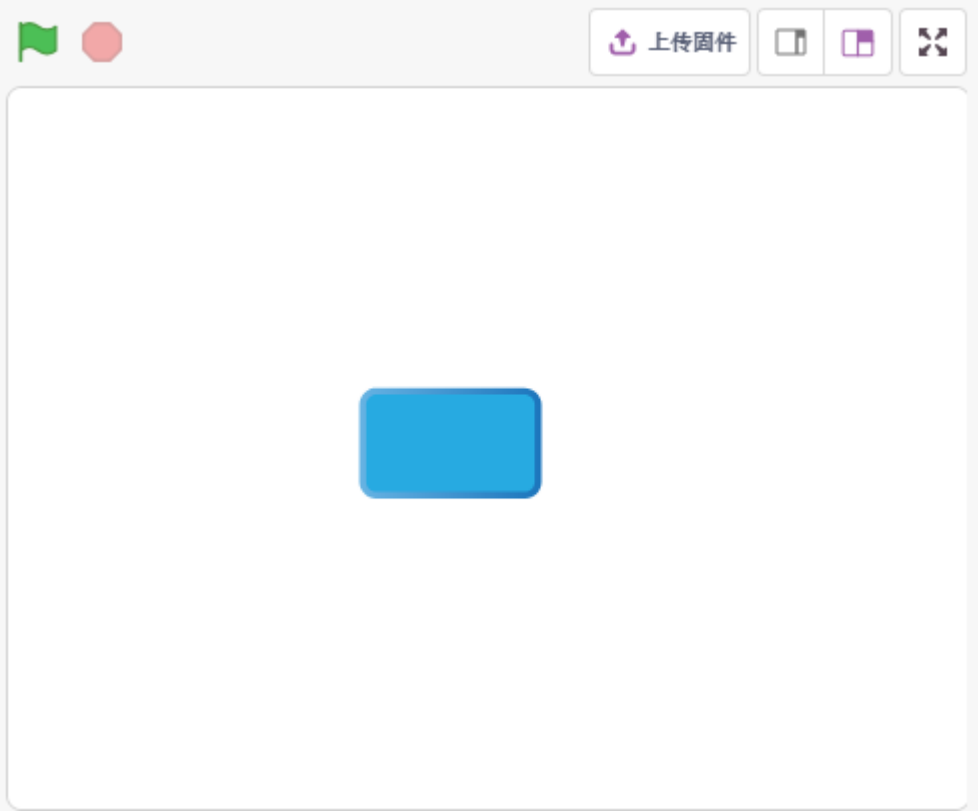
最后点击 **上传代码**按钮。


上传代码
当Arduino的乌诺启动
重复执行
组数字销 13 输出作为 低电平
等待 1 秒
组数字销 13 输出作为 高电平
等待 1 秒
//This arduino code is genera
void setup() {
//put your setup code here,
pinMode(13, OUTPUT);
}
void loop() {
//put your main code here,
digitalWrite(13, false);
delay(1 * 1000);
digitalWrite(13, true);
delay(1 * 1000);
}

8.3.2 2. 台灯

上一个项目是在让 Arduino 板上内置的 LED 点亮，这里，我们在面包板上接一个 LED，并由舞台上的精灵来控制这个 LED 的闪烁。

当舞台上的按键精灵被点击时，LED 将闪烁 5 次，然后停止。



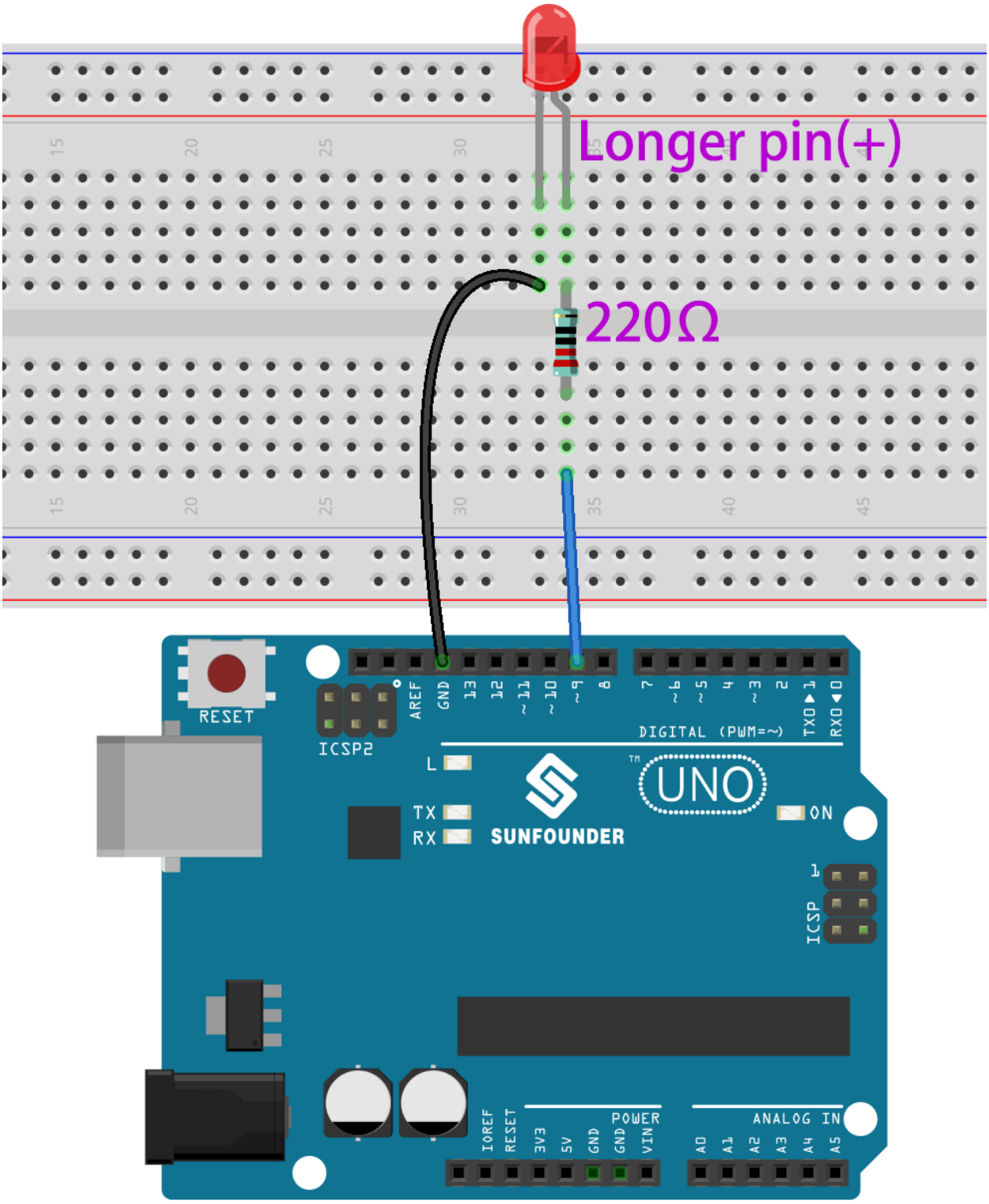
你将学习

- 面包板，LED 和电阻
- 在面包板上搭建电路
- 删除和选择精灵
- 切换造型
- 有限次数的循环

搭建电路

按照下图，在面包板上搭建电路。

由于 LED 的阳极（长的引脚）通过 220Ω 的电阻连接到 pin 9, LED 的阴极接 GND。所以你给 pin 9 高电平就能点亮这个 LED。



- 面包板
- *LED* 发光二极管
- 电阻

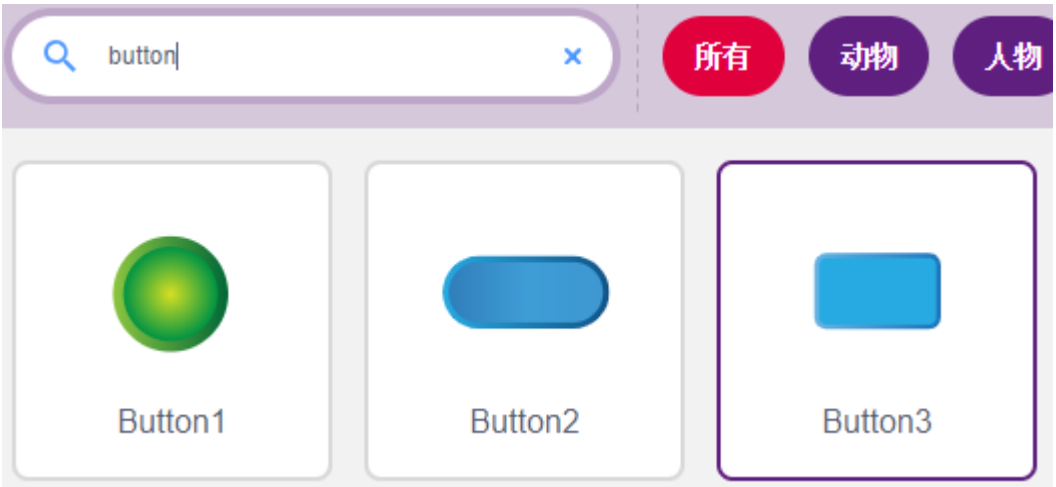
编程

整个编程分 3 个部分，第一个部分是选择想要的精灵，第二个部分是给精灵切换造型，让它看起来有点击的效果，第三个部分是让 LED 闪烁。

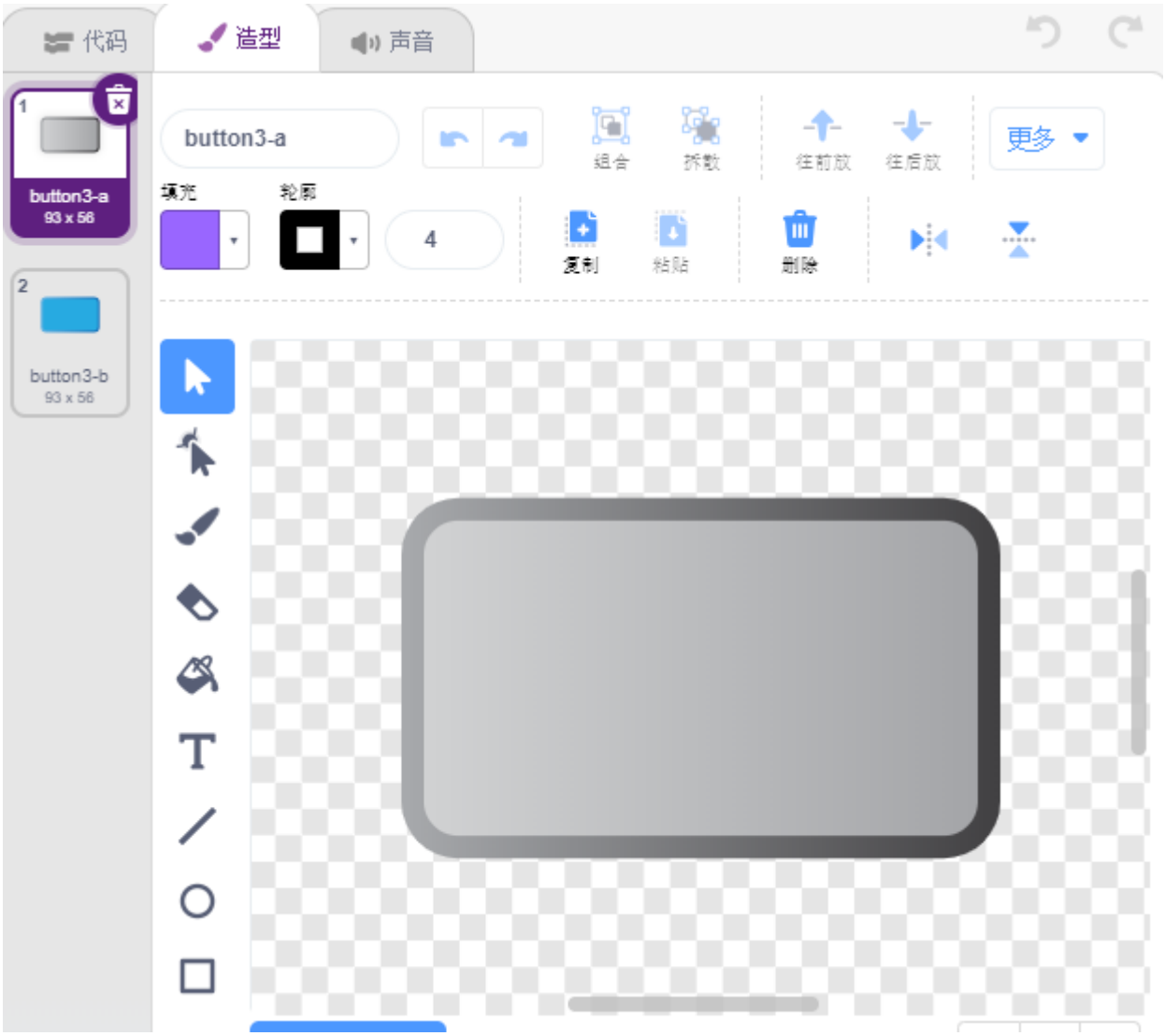
1. 选择 Button3 精灵

通过右上角的删除按钮来删除现有的 Tobi 精灵, 并重新选择一个精灵。

这里，我们选择 **Button3** 精灵。

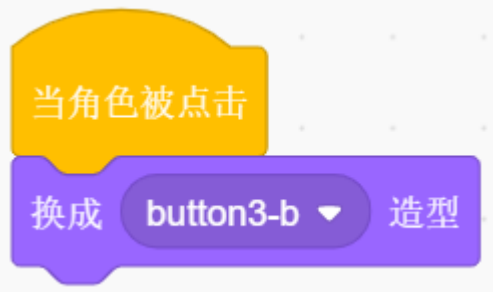


点击右上角的 **造型**，你会发现 Button3 精灵有 2 个造型, 我们设定 button3-a 为松开状态，button3-b 为按下状态。



2. 切换造型

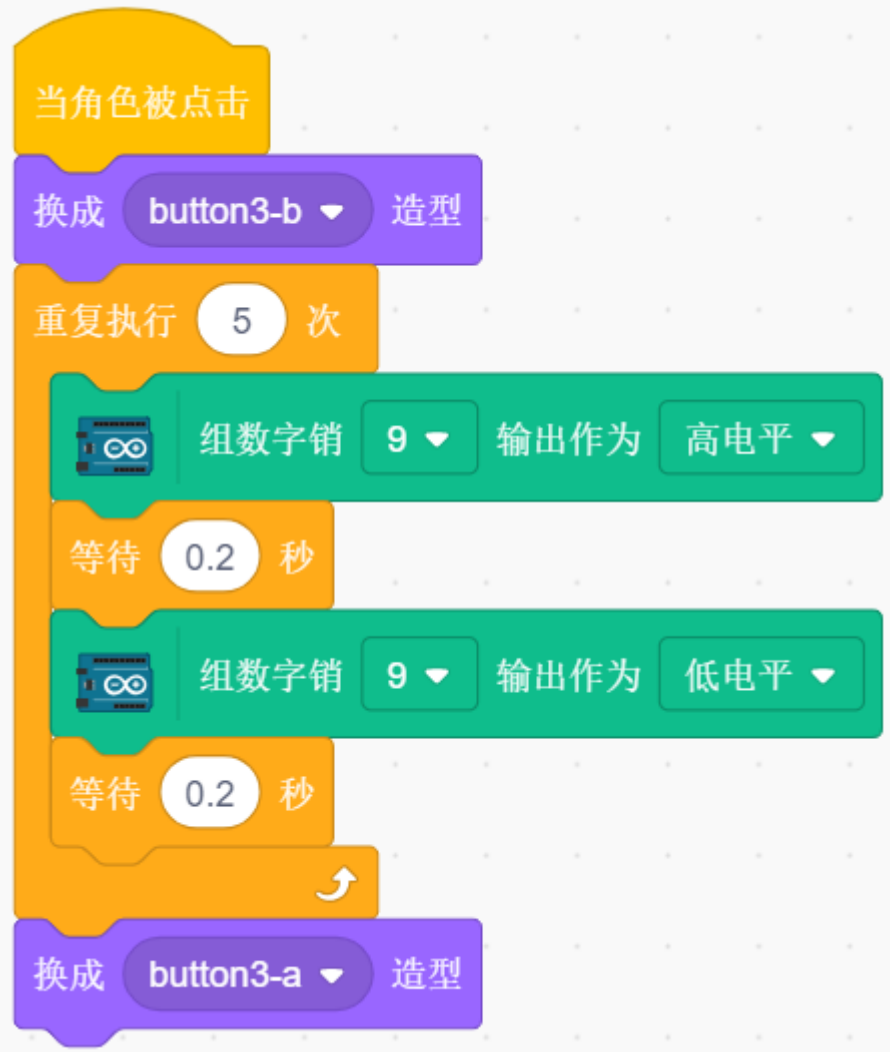
当精灵被点击（事件调色板），就切换到造型为 button3-b（外观调色板）。



3. 让 LED 闪烁 5 次

使用 [重复执行（次）] 块让 LED 闪烁 5 次 (高-> 低的循环)，记得 13 改为 9, 最后将造型切换回 button3-a。

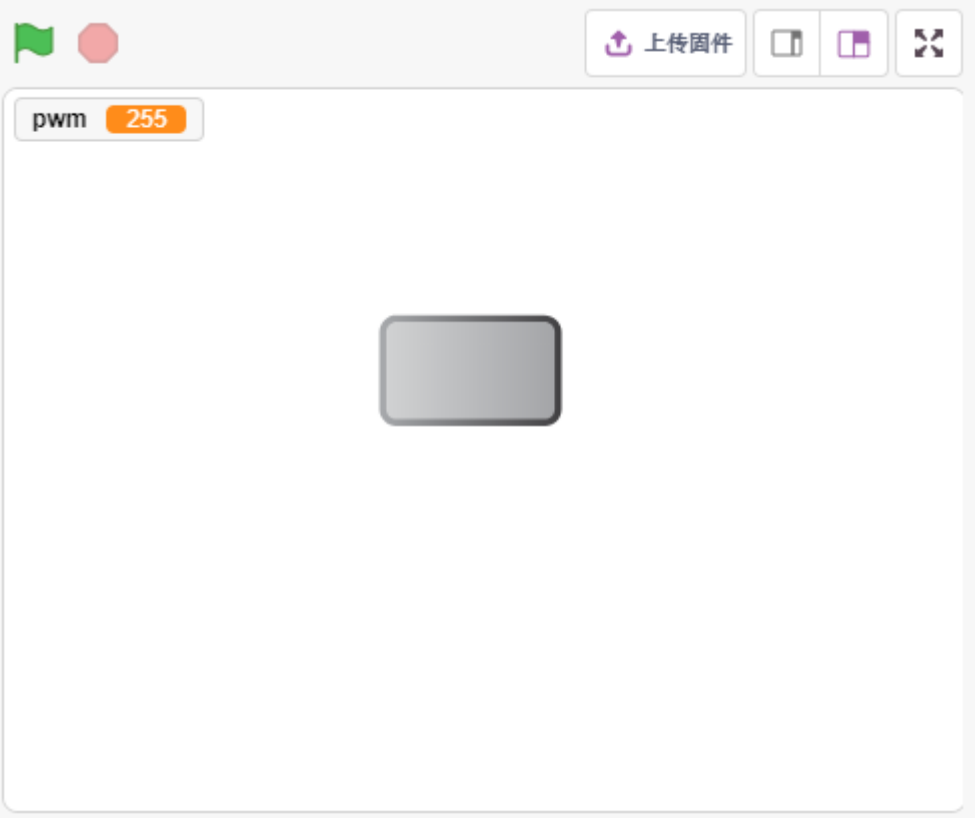
- [Repeat 10]: 重复 block, 可自己设定重复次数，来自 **控制**调色板。



8.3.3 3. 呼吸灯

现在用另外一种方法控制 LED 的亮灭，与上一个项目不同的是，这里是让 LED 的亮度慢慢的减弱，直到消失。

当舞台上的精灵被点击时，LED 的亮度慢慢增强，然后瞬间熄灭。



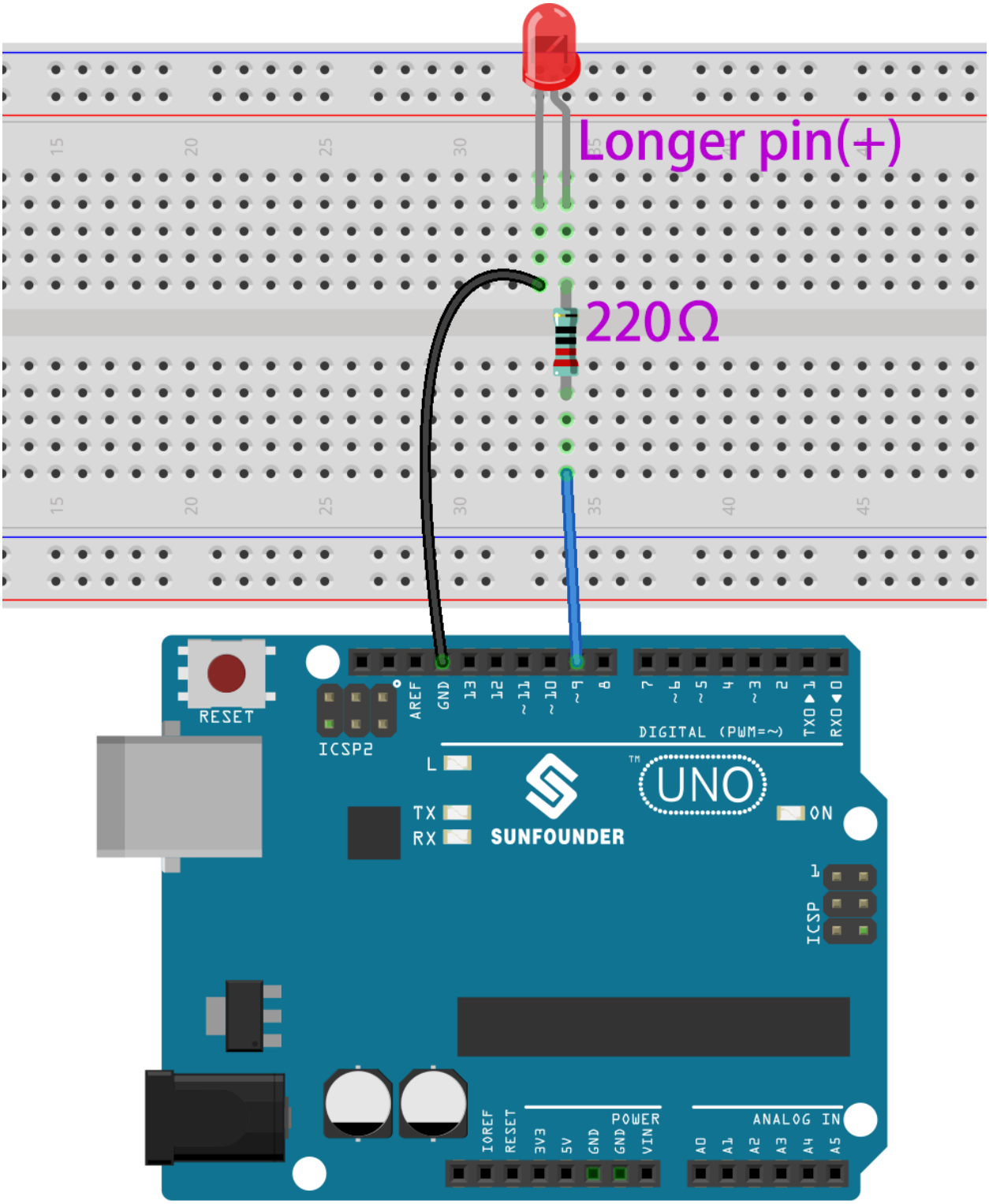
你将学习

- 设置 PWM 引脚的输出值
- 创建变量
- 改变精灵的亮度

搭建电路

这个项目用的还是上一个项目2. 台灯 的电路。但相比于上一个项目直接用 HIGH/LOW 让 LED 点亮或者熄灭，这个项目用的是 PWM 信号的方式来让 LED 慢慢变亮或者变暗。

PWM 信号值范围为 0-255，在 Arduno Uno 板上，3, 5, 6, 9, 10, 11 能够输出 PWM 信号；Mega2560 上，2 - 13, 44 - 46 能够输出 PWM 信号。



- 面包板
- *LED* 发光二极管
- 电阻

编程

1. 选择精灵

删除默认精灵，点击精灵区域右下角的选择精灵按钮，在搜索框中输入 **button3**，然后点击添加。

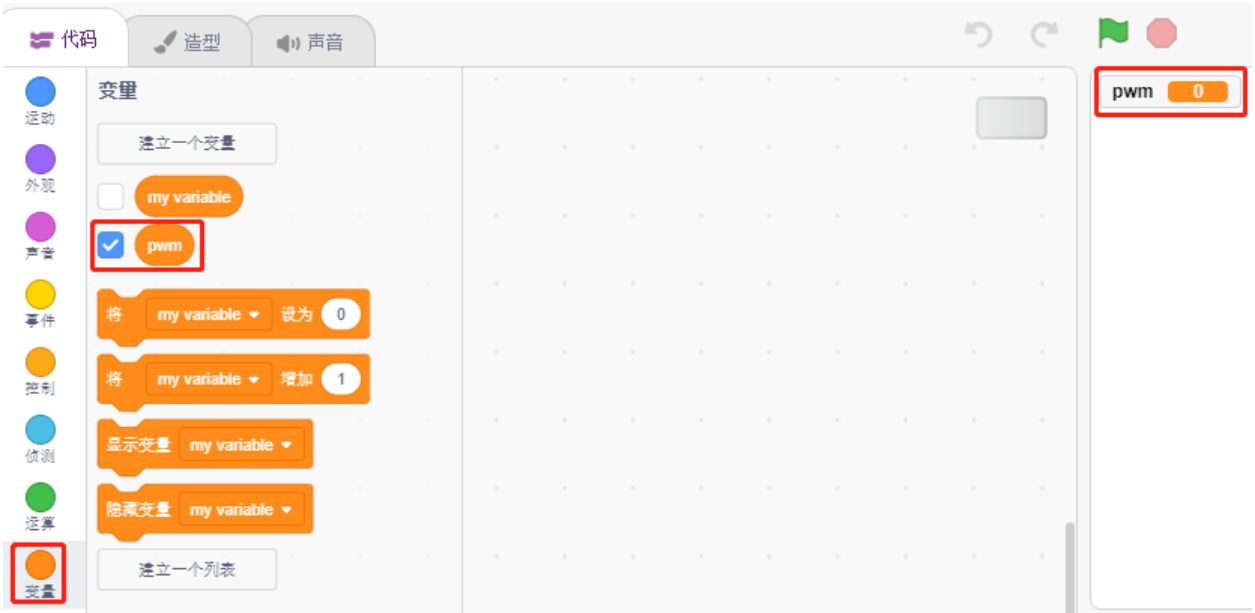
2. 创建变量

创建一个叫做 pwm 的变量，用来 pwm 变化的值。

点击 **变量**调色板，选择 **建立一个变量**。

输入变量名字，可以是任意名字，但建议能描述它的功能。数据类型为 number 和适用所有角色。

创建完成后，你会看到 **pwm** 出现在 **变量**调色板里面，并且处于勾选状态，这代表这个变量将出现在舞台上。你可以试着去掉勾选，看下舞台上 pwm 还是否存在。



3. 设置初始状态

当 button3 精灵被点击时，切换造型为 button-b(被点击状态)，并将变量 pwm 的初始值设置为 0.

- [将 () 设为 ()]: 来自变量调色板，用来设置变量的值。

4. 让 LED 越来越亮

由于 pwm 的范围是 255，所以通过 [重复执行 () 次] 块, 将变量 pwm 以 5 累加到 255，然后放到 [将 () 增加 ()] 块中，这样就能看到在上的 LED 慢慢点亮。

- [将 () 增加 ()]: 来自变量调色板, 让变量每次改变特定的数。可以是正数或负数，正数是每次增加，负数是每次减少，比如这里是让变量 pwm 每次增加 5.
- [组 PWM 引脚 () 输出作为 ()]：来自 Arduino 的乌诺调色板, 用来给 pwm 引脚设定输出值。

最后，将 button3 的造型切换回 button-a，并且让 PWM 引脚的值为 0，这样 LED 就会在慢慢点亮后又熄灭。

8.3.4 4. 彩灯

在这个项目中，我们将使 RGB LED 显示不同的颜色。

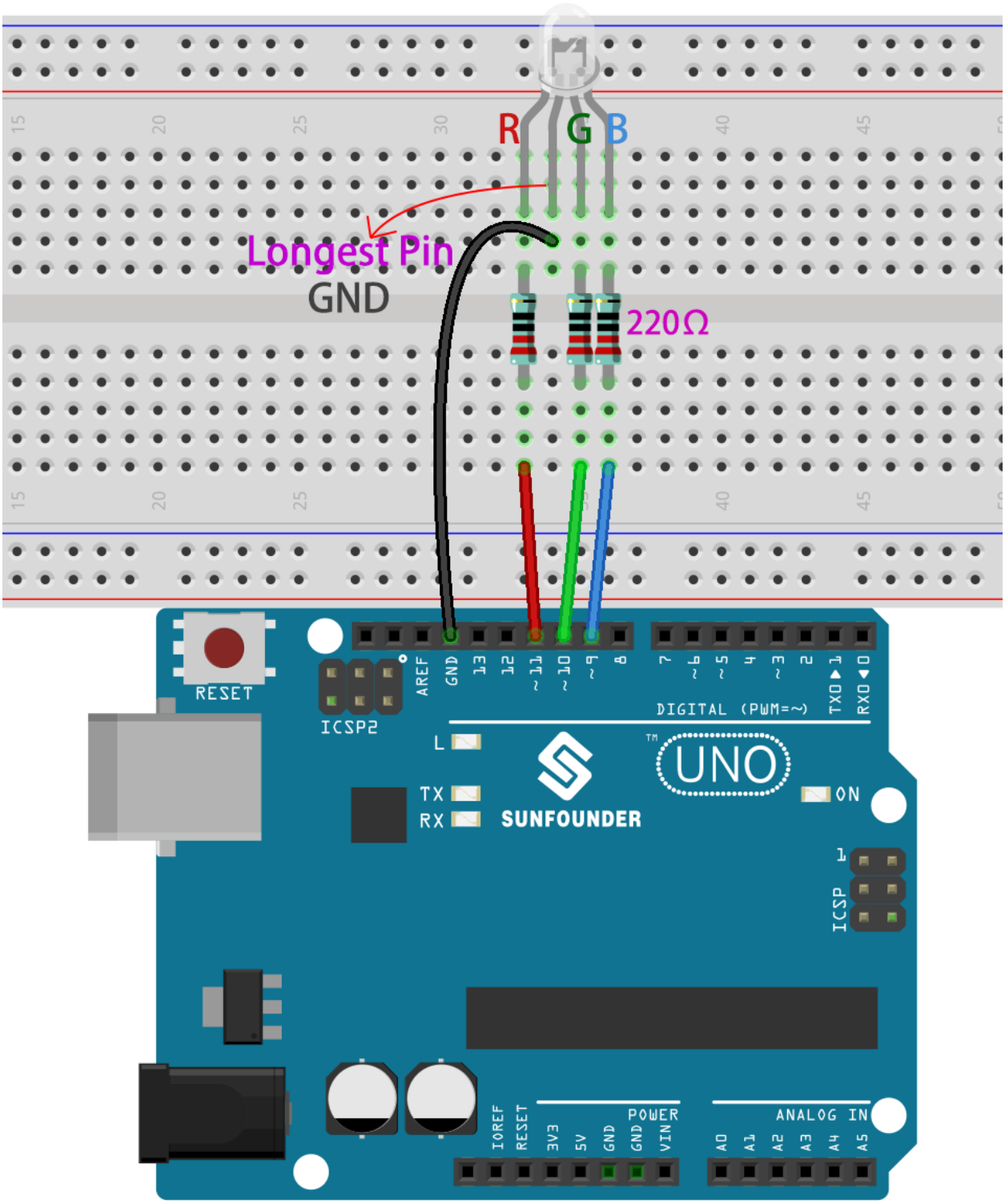
在舞台区域点击不同颜色的球会导致 RGB LED 显示不同颜色。

你将学习

- RGB LED 的原理
- 复制精灵及选择不同造型
- 三原色叠加

搭建电路

RGB LED 将红、绿、蓝三种 LED 封装在透明或半透明的塑料外壳中。它可以通过改变三个引脚的输入电压并叠加来显示各种颜色，据统计，可以产生 16,777,216 种不同的颜色。

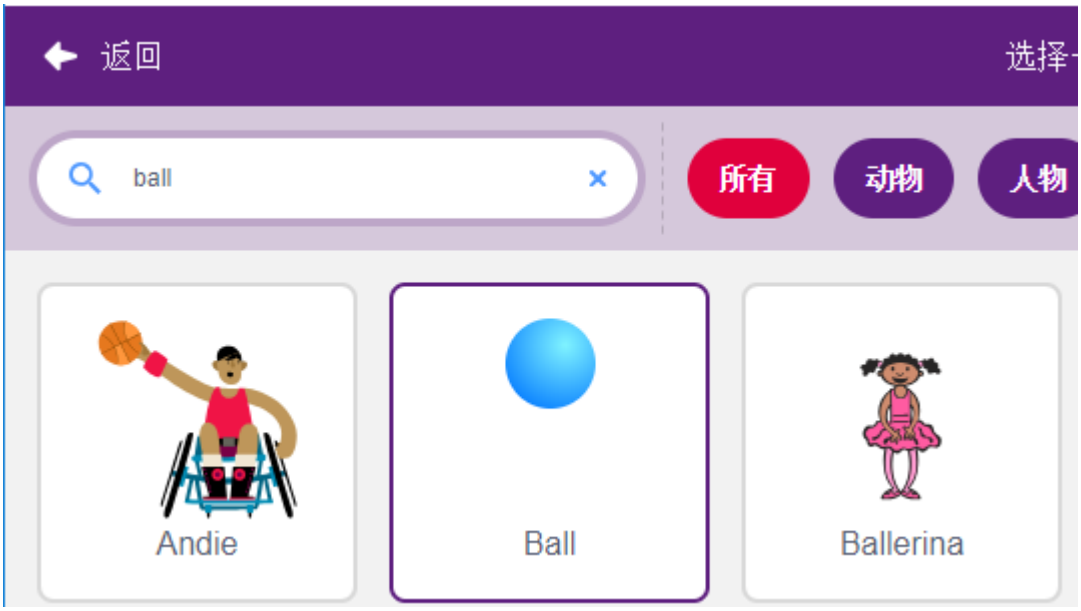


- 面包板
- 电阻
- *RGB LED*

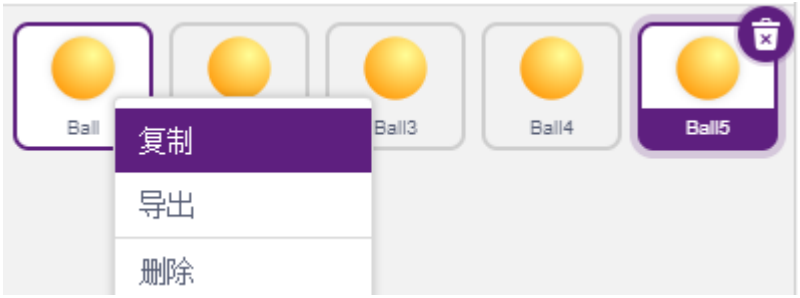
编程

1. 选择精灵

删除默认精灵，然后选择 **Ball** 精灵。

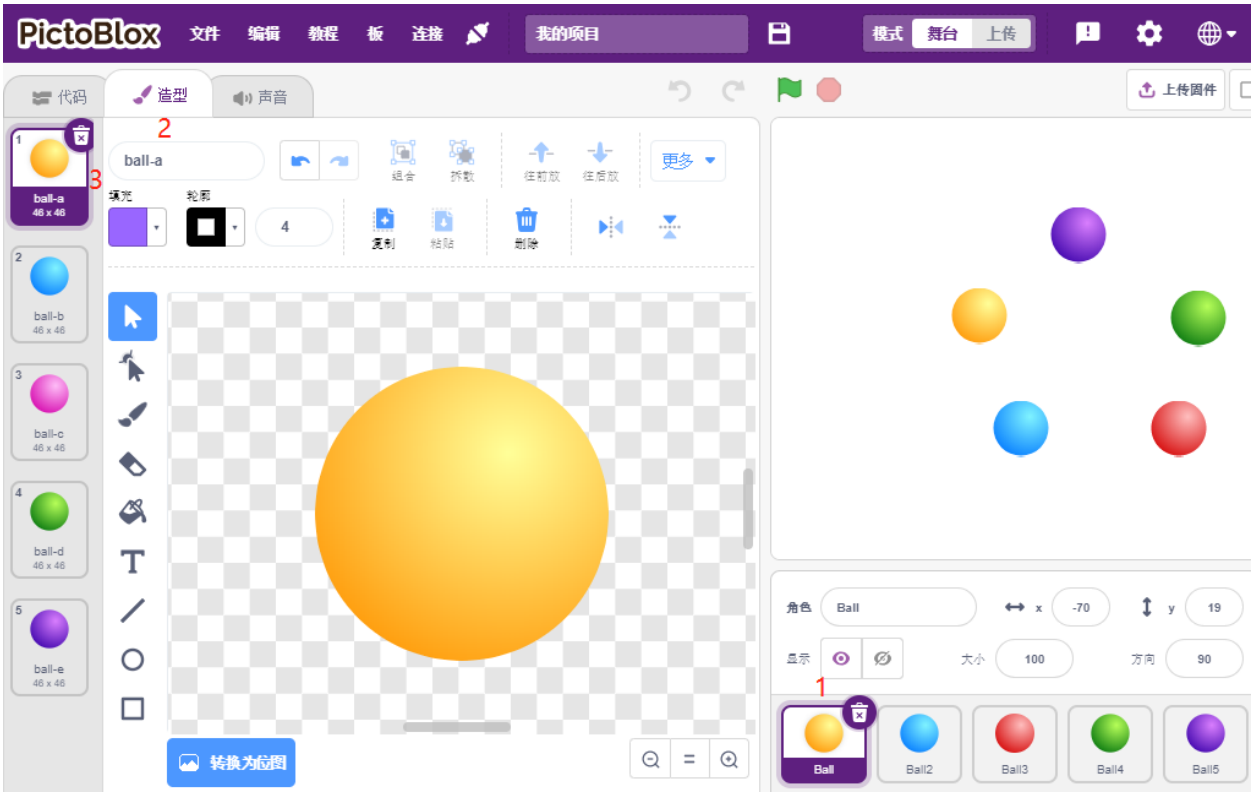


并复制 5 次



为这 5 个 **Ball** 精灵选择不同的造型并将它们移动到相应的位置。

备注： Ball3 的造型颜色需要手动改成红色。

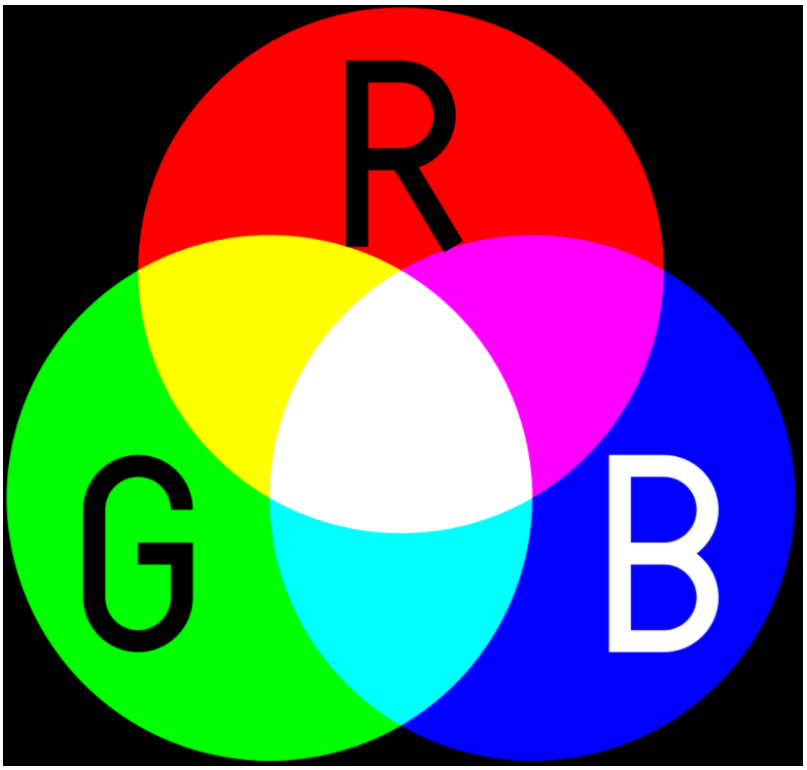


2. 让 RGB LED 点亮相应颜色

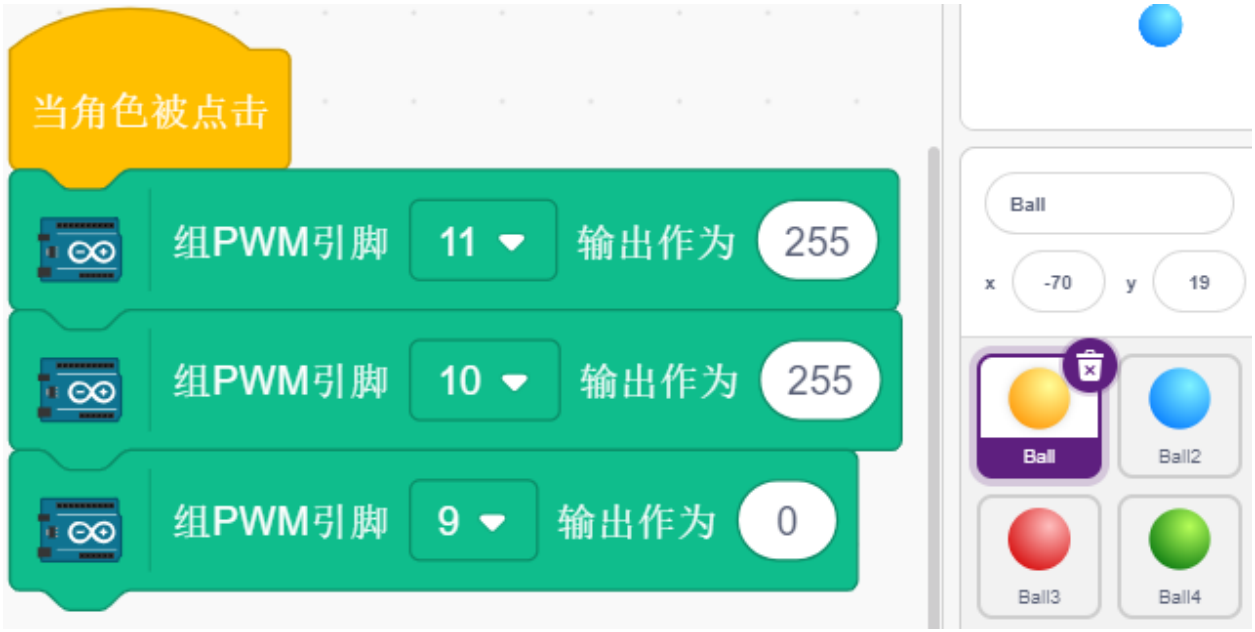
在理解代码之前，我们需要了解 RGB 颜色模型。

RGB 颜色模型是一种加色模型，其中红、绿和蓝光以各种方式叠加在一起，以再现各种颜色。

加色混色：红色加绿色产生黄色；将绿色添加到蓝色产生青色；在红色中加入蓝色会产生洋红色；将所有三种原色加在一起产生白色。



所以使 RGB LED 呈黄色的脚本如下。



当 **Ball** 精灵（黄球）被点击时，我们将引脚 11 设置为高电平（红色 LED 亮起）、引脚 10 高电平（绿色 LED 亮起）和引脚 9 低电平（蓝色 LED 关闭），以便 RGB LED 亮黄色。

你可以用同样的方法给其他精灵写脚本，让 RGB LED 亮起相应的颜色。

3. Ball2 精灵 (浅蓝色)

4. Ball3 精灵（红色）

5. Ball4 精灵（绿色）

6. Ball5 精灵（紫色）

8.3.5 5. 跳动的字符

LCD1602 可以用来显示 2x16 个字符，现在我们让它随着舞台上的字符精灵来显示相应的字符。

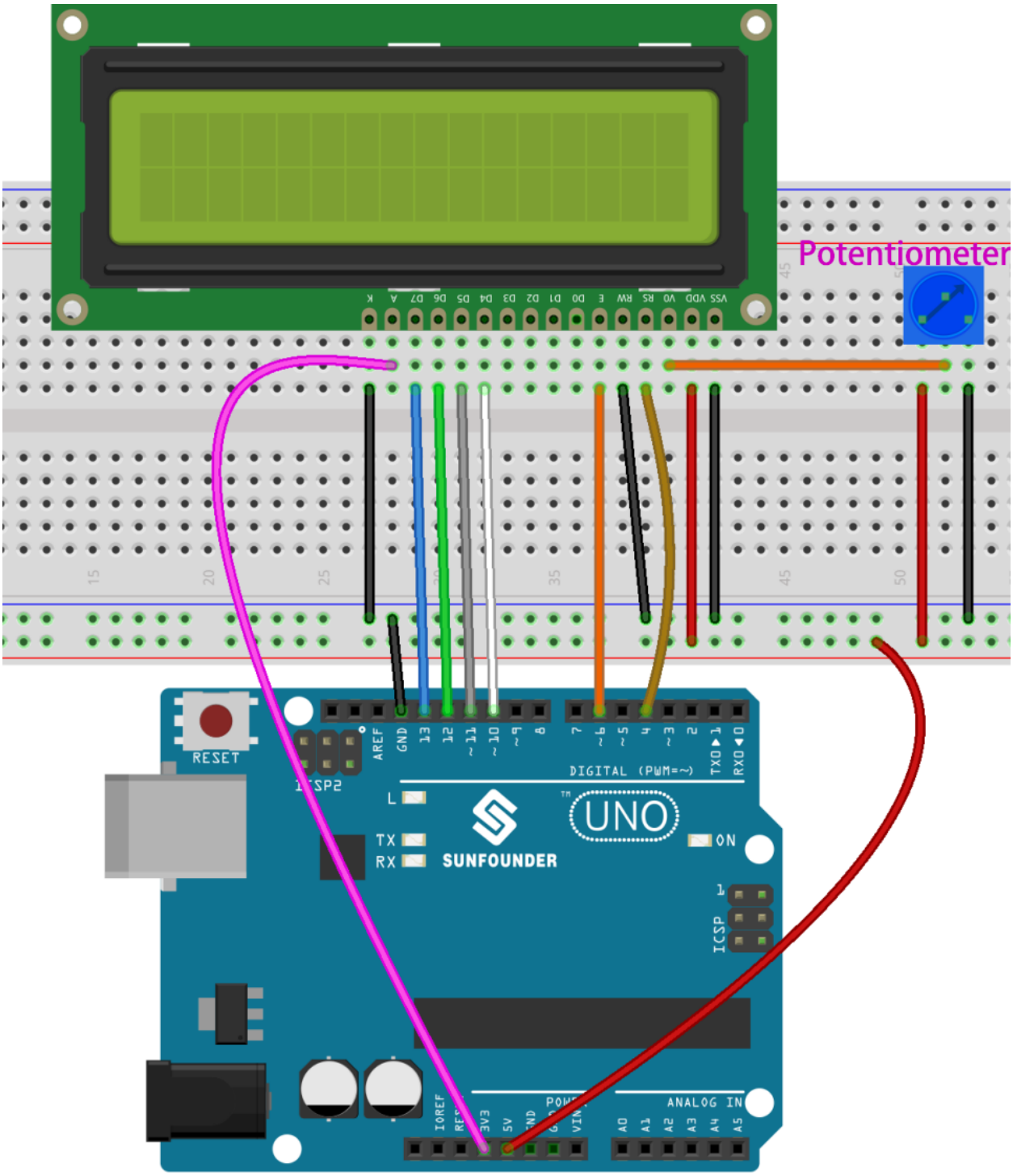
当你一一点击舞台上的 Hello 时，它们会有不同的动画效果，LCD1602 上会同时显示这些字符。

You Will Learn

- 使用 LCD1602
- 选择多个不同的精灵
- 改变精灵尺寸，转动角度，颜色和显示或隐藏。

搭建电路

- 连接 **K** 到 GND 和 **A** 至 3.3 V，然后将 LCD1602 的背光将被打开。
- 将 **VSS** 连接到 GND。
- 将 **VO** 连接到电位器的中间引脚 - 你可以使用它来调整屏幕显示的对比度。
- 连接 RS 到 D4 和 R / W 连接到 GND，该装置则可以写入字符的 LCD1602。
- 将 **E** 接 6 引脚，LCD1602 上显示的字符由 **D4-D7** 控制。

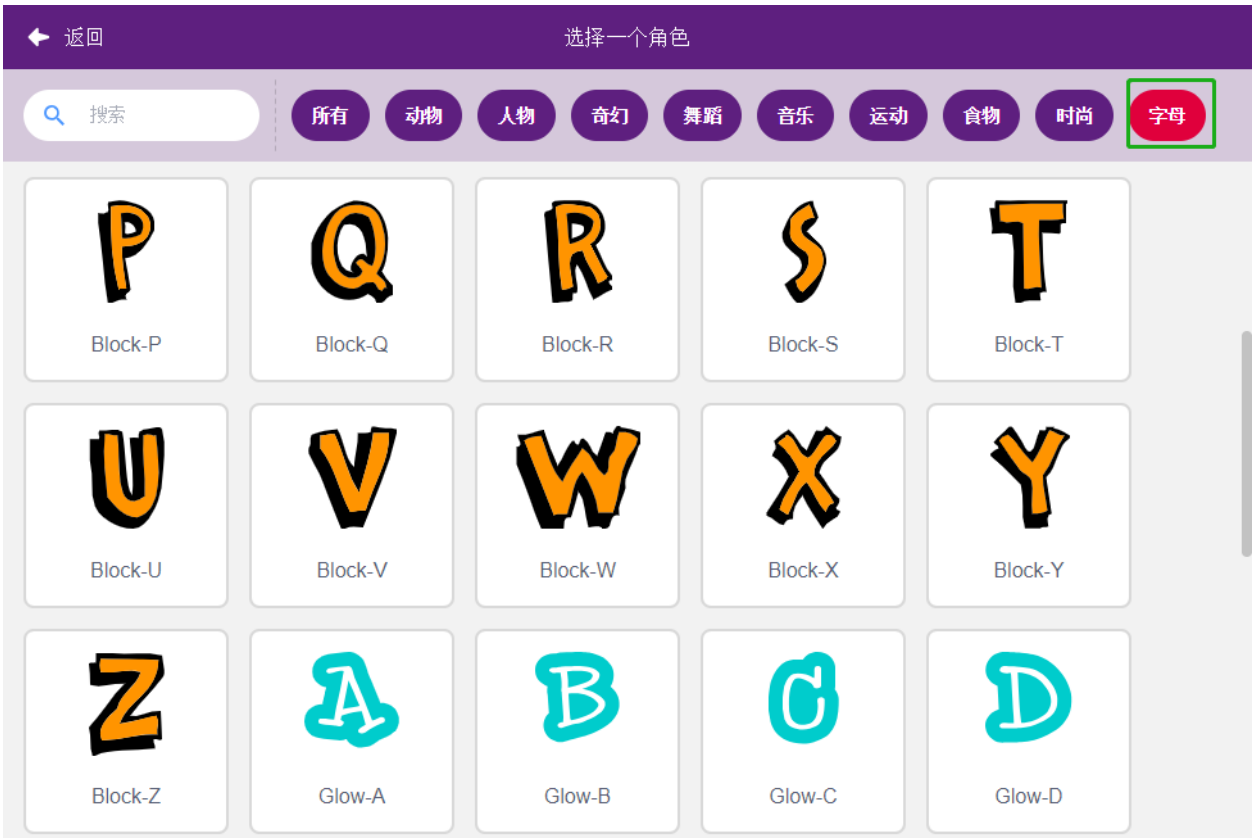


- 面包板
- *LCD1602* 液晶显示屏
- 电位器

编程

1. 选择精灵

删除默认精灵，点击 **选择一个角色**, 然后点击 **字母**之后选择你想要的字符精灵。



比如我选择的是 Hello，如下所示。

现在来给这些精灵设置不同的效果，并在点击的同时显示在 LCD1602 上。

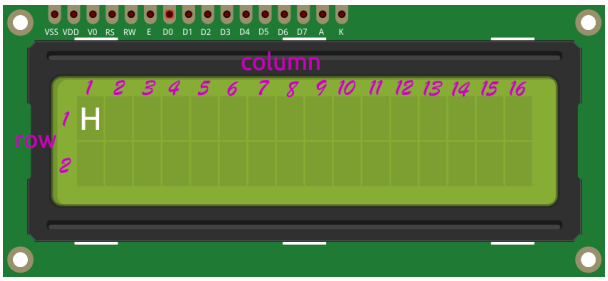
2. H 变大缩小

点击 **H** 精灵，现在来为它编写脚本：

当精灵 H 被点击时，让它的大小缩小成 50%，然后恢复，同时在 LCD1602 的第一行列和第一列显示 H。

- [将大小设为 ()]: 来自外观调色板，用来设置精灵的尺寸，范围是 0%-100%。
- [在列集光标 () 行 ()]: 来自显示模块调色板，用来将光标设定在 LCD1602 特定的行列，以此开始显示字符。
- [写 () 显示]: 来自显示模块调色板，用来在 LCD1602 上显示字符或字符串。

LCD1602 上的行列显示原理如图所示。



3. E 在左右翻转

点击 **E** 精灵，现在来为它编写脚本：

当精灵 E 被点击时，让它顺时针转 180 度，然后逆时针转 180 度，这样你就能看到它左右翻转，同时在 LCD1602 的第一行和第 2 列显示 H。

- [左转/右转 () 度]：来自运动调色板，用来顺时针或逆时针转动精灵角度，范围为 0-360 度。

4. L 在慢慢缩小和放大

点击第一个 L 精灵，现在来为它编写脚本：

当精灵 L 被点击时，以每次增加 10 的速度，使用 [重复执行 () 次] 块重复 5 次，将它的大小增加 50，然后按照同样方法缩小回原来大小, 同时在 LCD1602 的第一行和第 3 列显示 L。

- [将大小增加 ()]: 来自运动调色板，用来改变精灵的尺寸。

5. 第二个 L 在改变颜色

点击第二个 L 精灵，现在来为它编写脚本：

当精灵 L 被点击时，以每次增加 20 的速度，使用 [重复执行 () 次] 块重复 10 次，让它切换不同的颜色并回到原始的颜色. 同时在 LCD1602 的第一行和第 4 列显示 L。

- [将颜色特效增加 ()]: 用于改变颜色效果，一套造型使用该颜色效果可以呈现 200 种不同的配色方案，0 和 200 是相同的颜色。

6. O 在显示和隐藏

点击 O 精灵，现在来为它编写脚本：

当精灵 O 被点击时，它重复 3 次隐藏和显示过程，同时在 LCD1602 的第一行和第 5 列显示 L。

- [隐藏] & [显示]：让精灵隐藏和显示。

8.3.6 6. 移动的老鼠

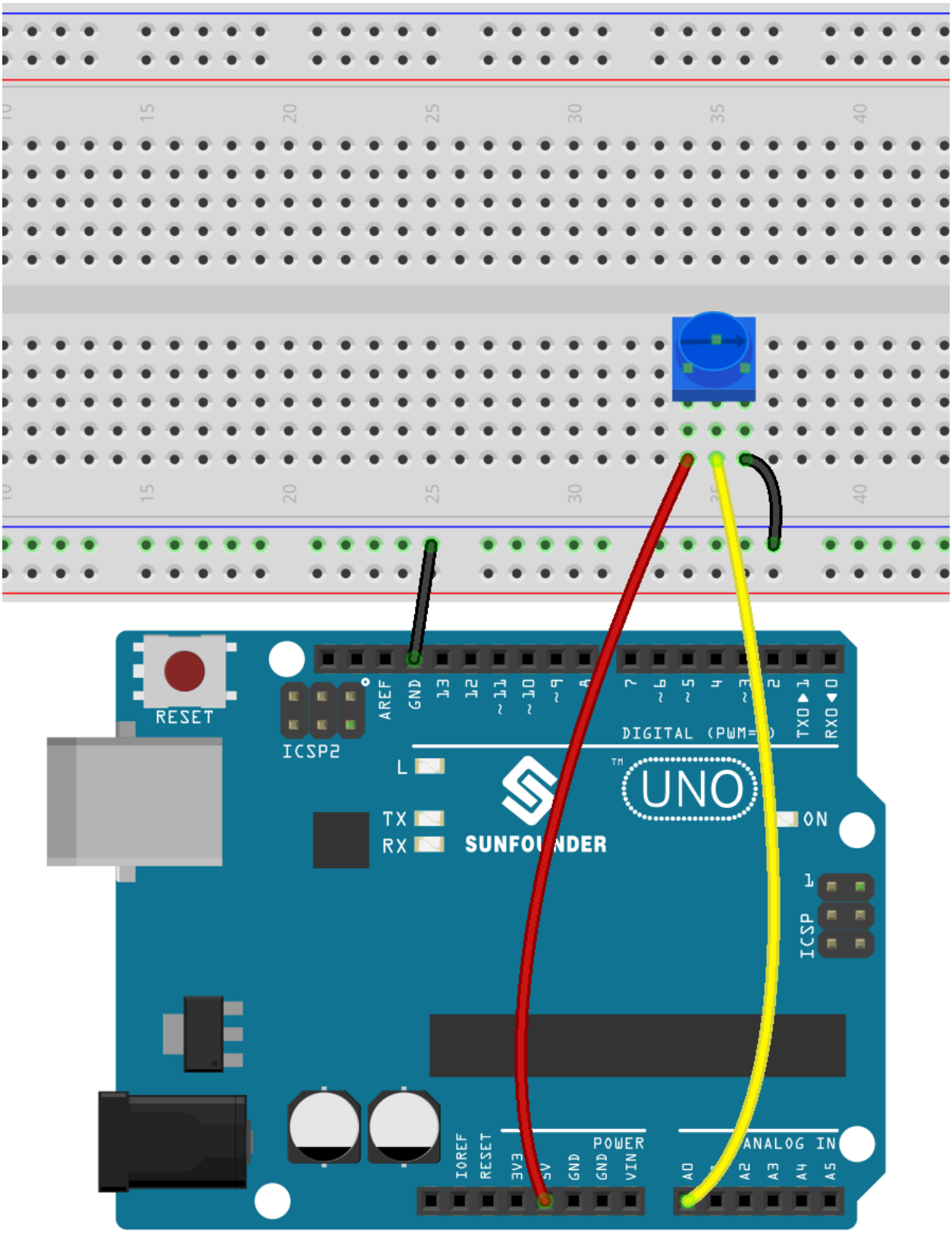
今天我们来制作一个由电位器控制的老鼠玩具。
当绿旗被点击时，舞台上的老鼠向前移动，当你旋转电位器，老鼠会改变移动方向。

你将学习

- 电位器原理
- 读模拟引脚及范围
- 将一个范围映射到另外一个范围
- 移动和改变精灵方向

搭建电路

按照下图搭建电路，电位器是一个有 3 个端子的电阻元件，2 侧的引脚分别接 5V 和 GND，中间引脚接到 A0，经过 Arduino 板的 ADC 转换器转换后，得到的数值范围为 0-1023.


RESET
ICSP2
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
1
0
DIGITAL (PWM=~)
TX
RX
L
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A2
A3
A4
A5
ANALOG IN

- 面包板

- 电位器

编程

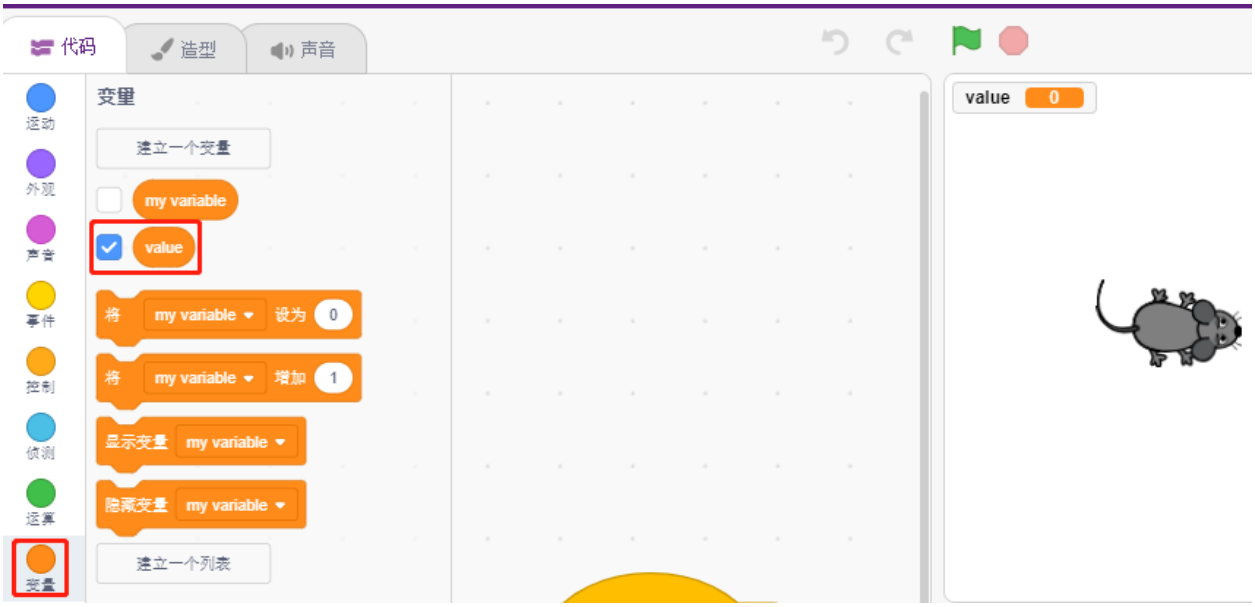
1. 选择 mouse 精灵

删除默认精灵，点击精灵区域右下角的选择精灵按钮，在搜索框中输入 **mouse**，然后点击添加。

2. 创建变量

创建一个叫做 value 的变量，用来存放读到的电位器的值。

创建完成后，你会看到 **value** 出现在变量调色板里面，并且处于勾选状态，这代表这个变量将出现在舞台上。



3. 读取 A0 的值

将读取的 A0 的值存到变量 value 里面。

- [将 () 设为 ()]: 设置变量的值。

- [读取模拟销 ()]: 读取 A0~A5 的值，范围为 0-1023。

为了能够一直读取，需要用到 [重复执行] 块. 点击这个脚本来将它运行，两个个方向旋转点位器，你会发现 value 的值范围为 0-1023.

4. 移动这个精灵

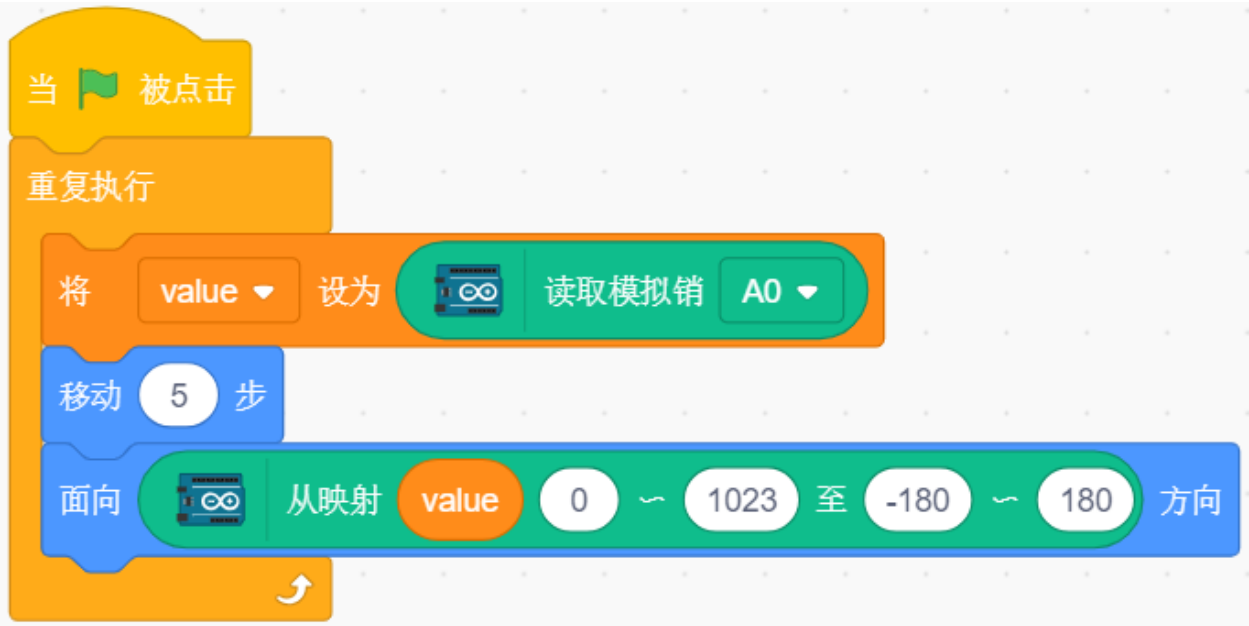
使用 [移动 () 步] 块让精灵移动，运行脚本，你会看到精灵从中间移到右边。

5. 改变精灵方向

现在通过 A0 的值来改变精灵的移动方向。由于 A0 的值范围是 0-1023，但精灵的转动方向为-180~180，所以需要用到一个 [从映射 () () ~ () 至 () ~ ()] 块。

另外在脚本开始添加 [当 () 被点击] 来启动这个脚本。

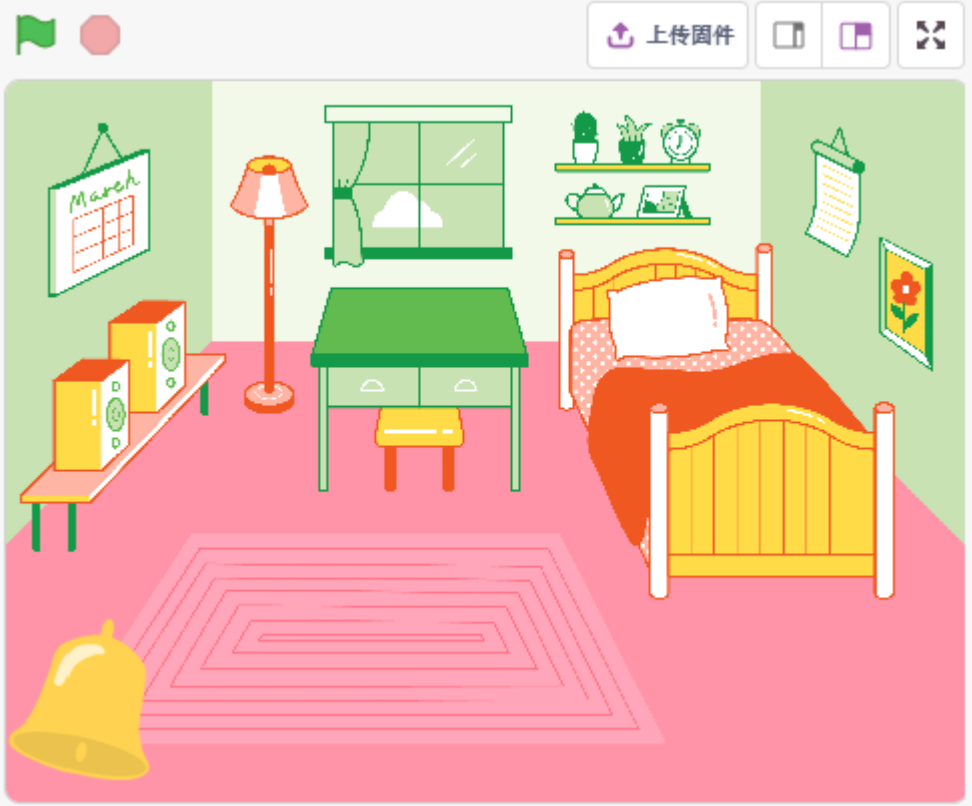
- [面向 () 方向]：设置精灵的转向角度，来自运动调色板。
- [从映射 () () ~ () 至 () ~ ()]: 将一个范围映射到另外一个范围。



8.3.7 7. 门铃

在这里，我们将使用按键和舞台上的铃铛来制作一个门铃。

当绿旗被点击时，你按下按键，铃铛精灵就会发出声音。



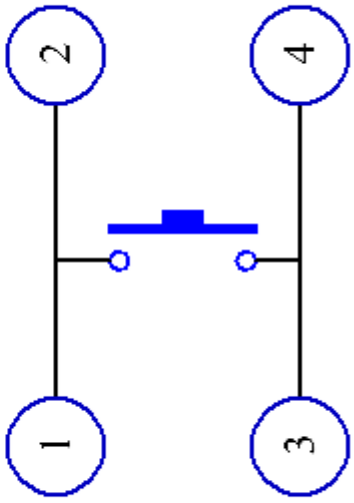
You Will Learn

- 按键工作原理
- 读数字引脚及范围
- 创建条件判断循环
- 添加背景
- 播放声音

搭建电路

该按钮为 4 引脚的器件，因为引脚 1 连接到引脚 1，引脚 3 连接到引脚 4，当按下按钮时，4 个引脚都连接在一起，从而闭合电路。

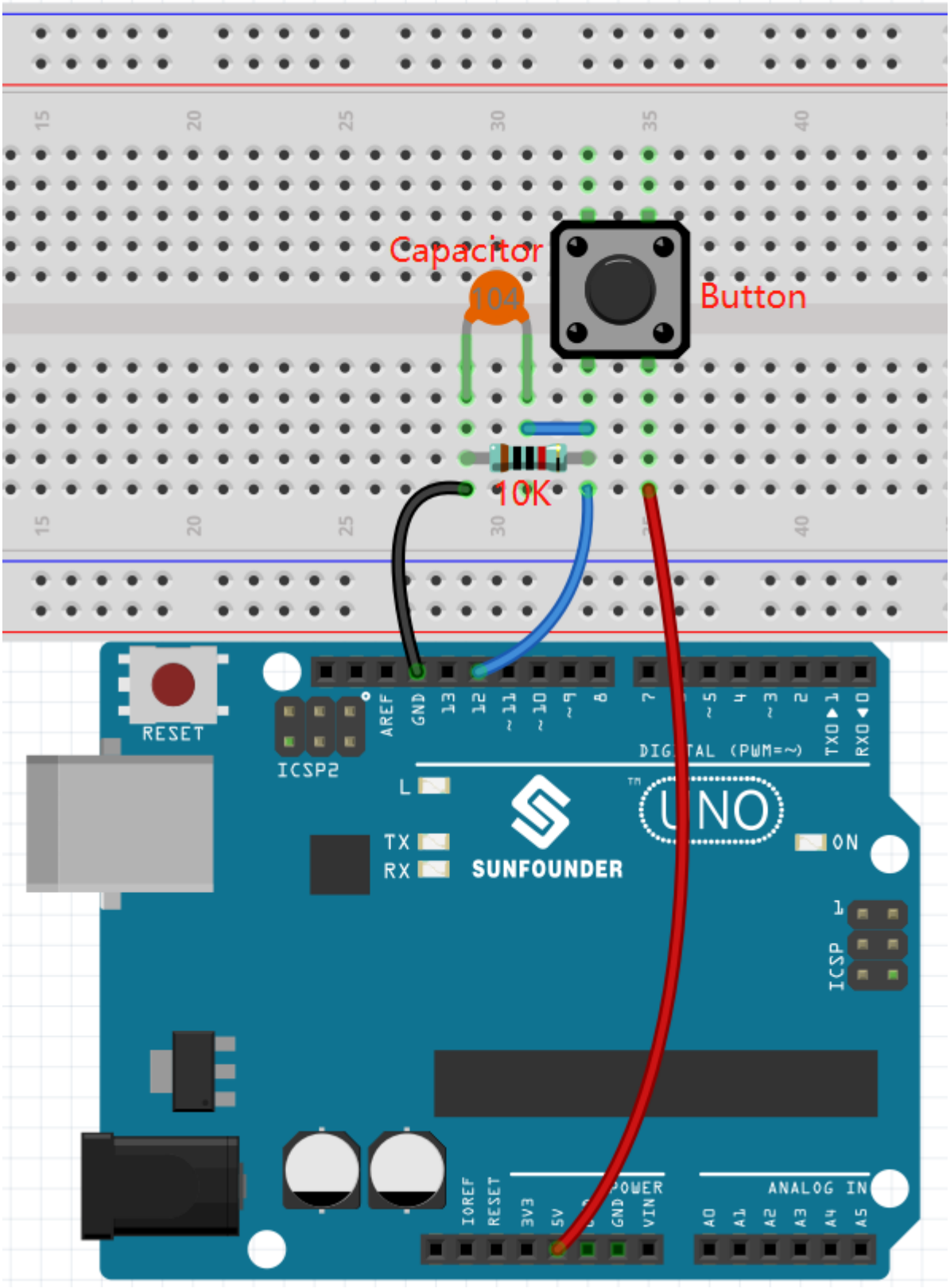
Button



Internal Structure

按照下图搭建电路：

- 将按钮左侧的其中一个引脚连接到 12 引脚，该引脚连接下拉电阻和 0.1uF（104）电容（以消除抖动并在按钮工作时输出稳定电平）。
- 将电阻和电容的另一端连接到 GND，将按钮右侧的一个引脚连接到 5V。


Capacitor
104
Button
10K
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

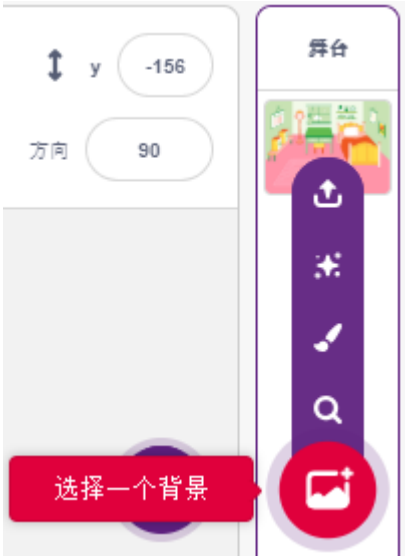
- 面包板
- 按键

- 电阻
- 电容

编程

1. 选择背景

点击右下角的 **选择一个角色**按键。



选择一个室内的背景，比如选择 **Bedroom 1**.

2. 选择精灵

删除默认精灵，点击精灵区域右下角的 **选择一个角色**按钮，在搜索框中输入 bell，然后点击添加。

然后选择舞台上的 bell 精灵，将它移动到合适的位置。

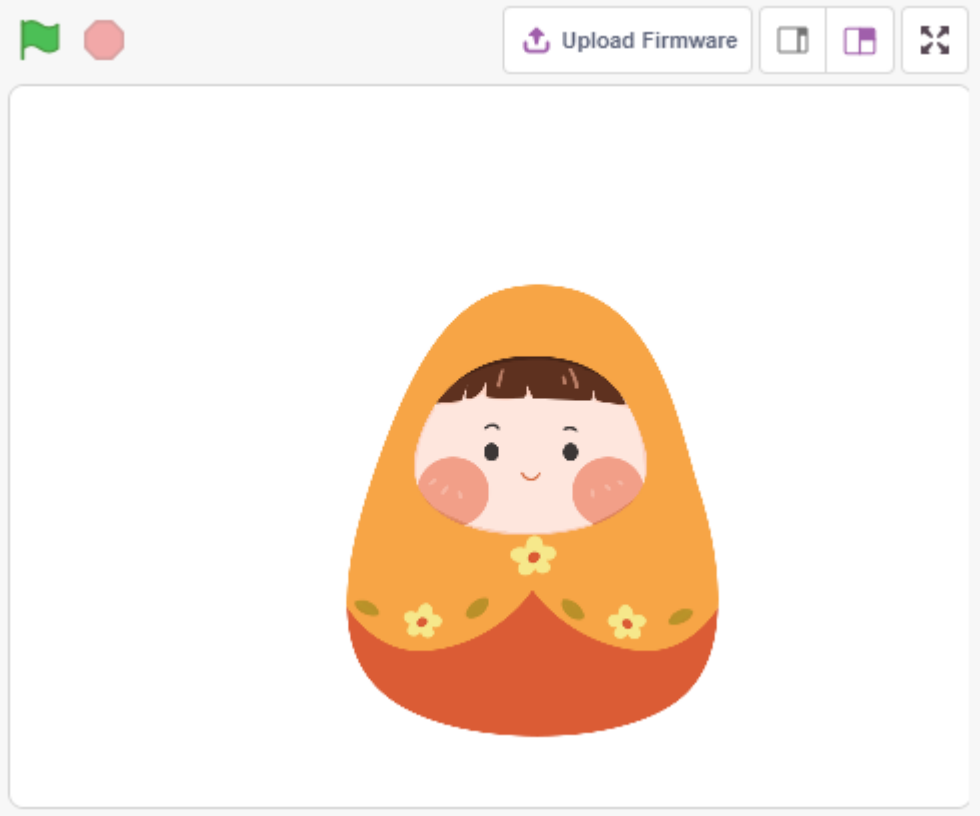
3. 按键按下，bell 发出声音

使用 [如果 <> 那么] 做一个判断语句，当读取的数字 12 的值等于 1，即按键按下，则播放声音 **xylo1**.

- [读出的数字销的状态 <>]: 来自 Arduino 的乌诺调色板，用于读取数字引脚的值，结果为 0 或 1.
- [如果 <> 那么]: 来自控制调色板. 如果它的布尔条件为真，它里面的块就会运行，然后所涉及的脚本将继续。如果条件为假，则块内的脚本将被忽略。条件只检查一次；如果在块内的脚本运行时条件变为假，它将继续运行直到完成。
- [播放声音 <> 等待播完]: 来自声音调色板, 用于播放特定的声音。

8.3.8 8. 不倒翁

现在我们用一个倾斜开关控制舞台上的不倒翁，让开关倾斜，不倒翁也倾斜。



You Will Learn

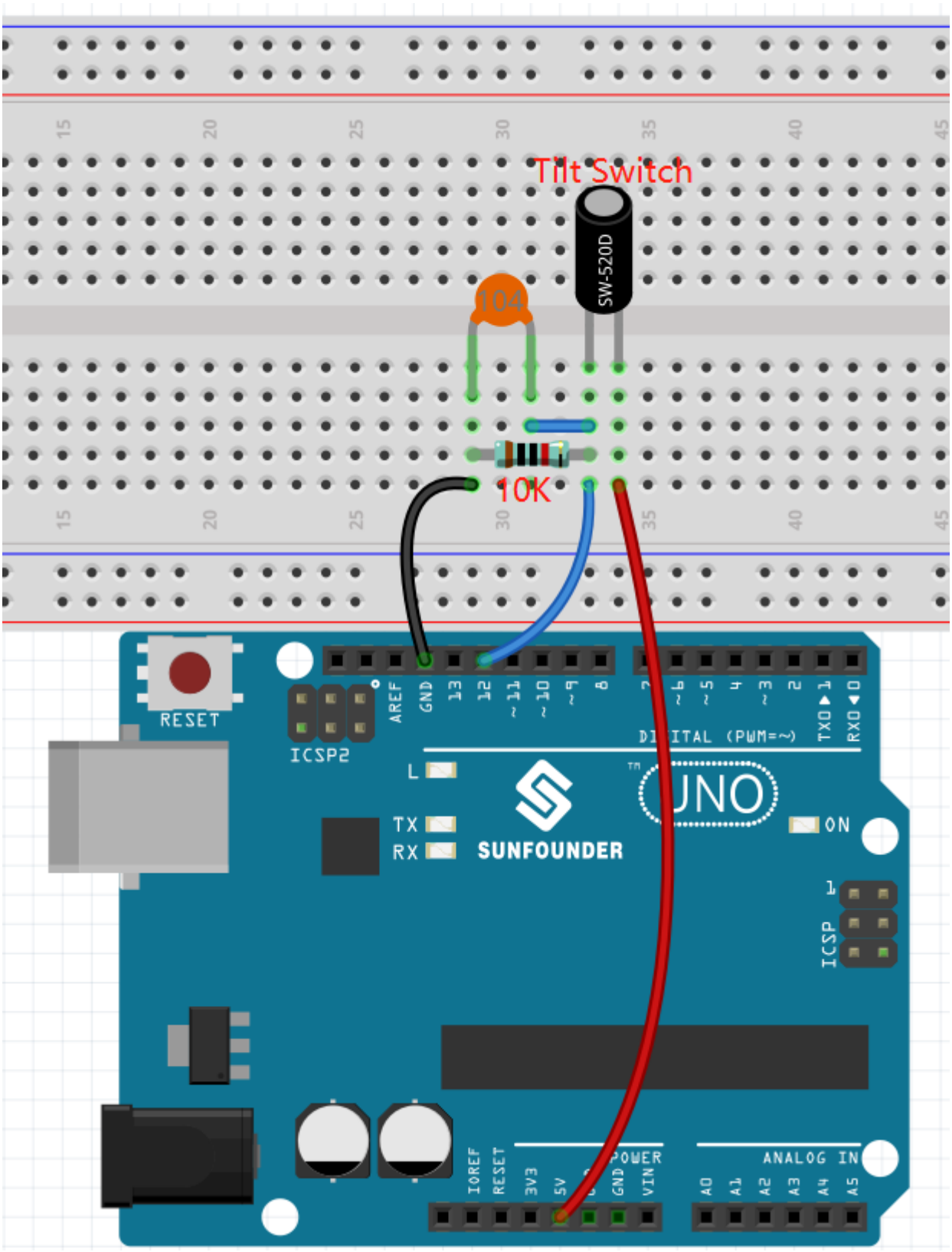
- 倾斜开关工作原理
- [如果 () 那么 () 否则 ()] 块
- 添加外部精灵

搭建电路

这里使用的倾斜开关是一个里面有金属球的器件。直立时，2 个引脚连在一起，倾斜时，它们是分开的。

根据下图搭建电路：

- 将倾斜开关的一个引脚连接到引脚 12，该引脚连接下拉电阻和 0.1uF（104）电容（用于在倾斜开关工作时消除抖动并输出稳定电平）。
- 将电阻和电容的另一端连接到 GND，将倾斜开关的另一个引脚连接到 5V。


Tilt Switch
SW-520D
104
10K
RESET
ICSP2
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TXD
RXD
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

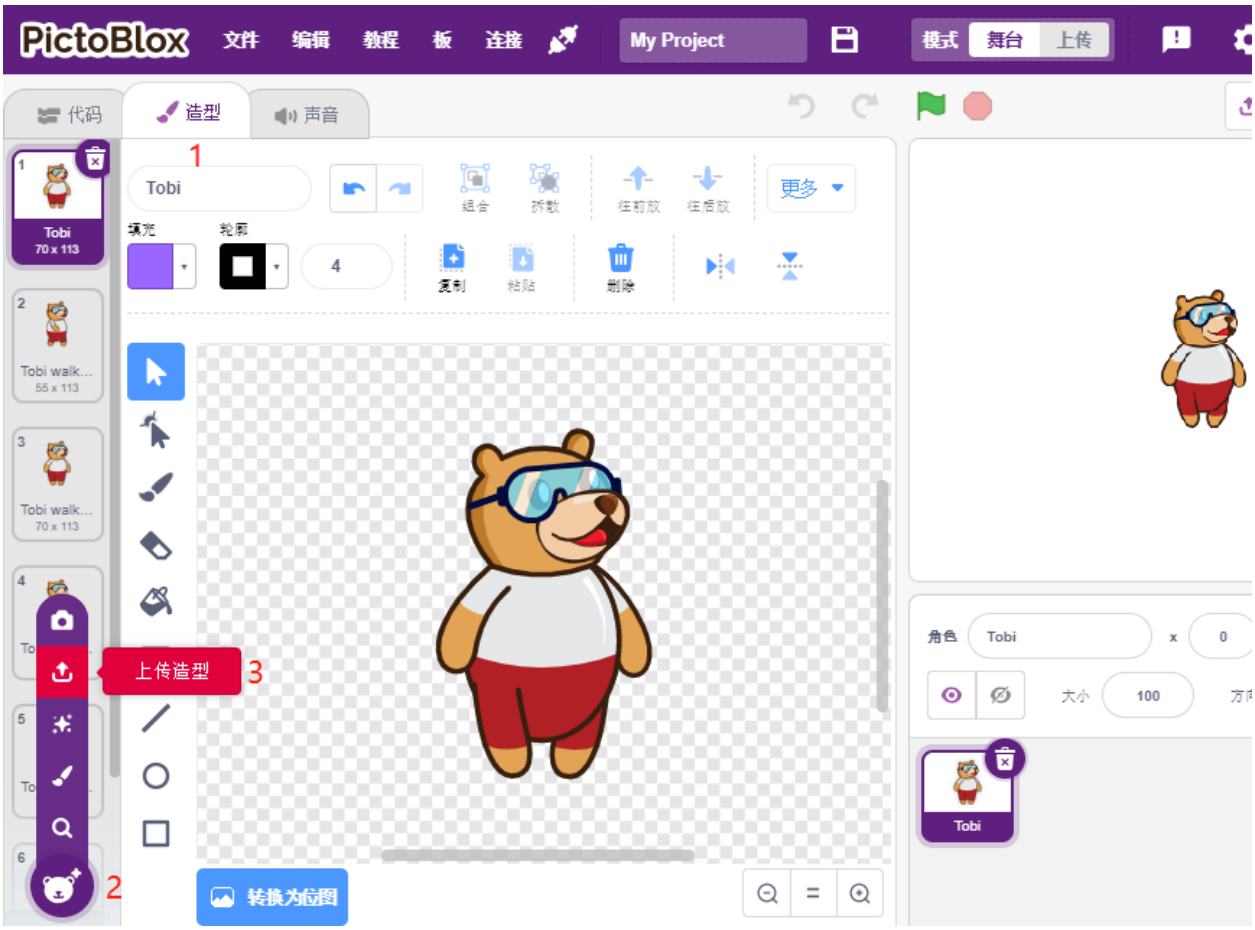
- 面包板
- 倾斜开关

- 电阻
- 电容

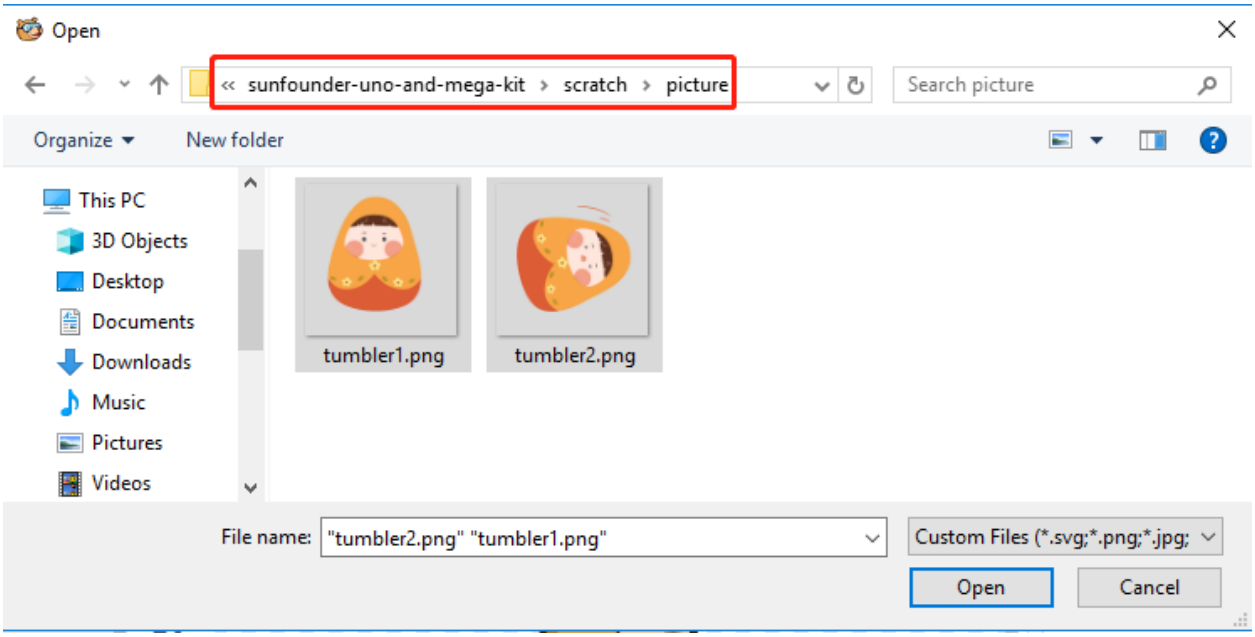
编程

1. 自定义精灵

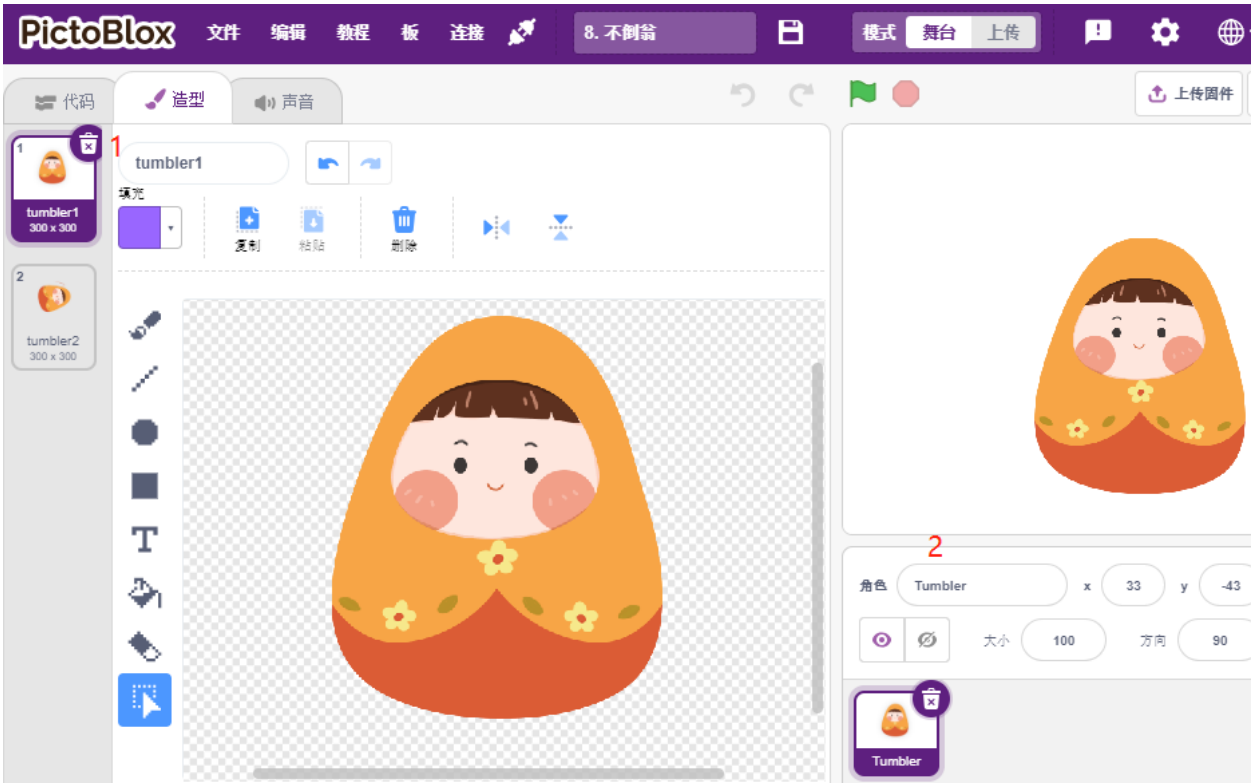
选中 Tobi 精灵，进入到造型页面中。选中左下角的图标，然后选择 **上传造型**。



同时选中 `SunFounder Uno R3 学习套件\Scratch 项目代码\图片路径中的 tumbler1.png` and `tumbler2.png`, 请确保你已经参考下载资料 下载了相关资料。



删除 Tobi 精灵相关的造型，并将名字改为 **Tumbler**。现在我们已经定制了一个新的精灵 Tumbler，现在开始为它编写脚本。



2. 让开关倾斜

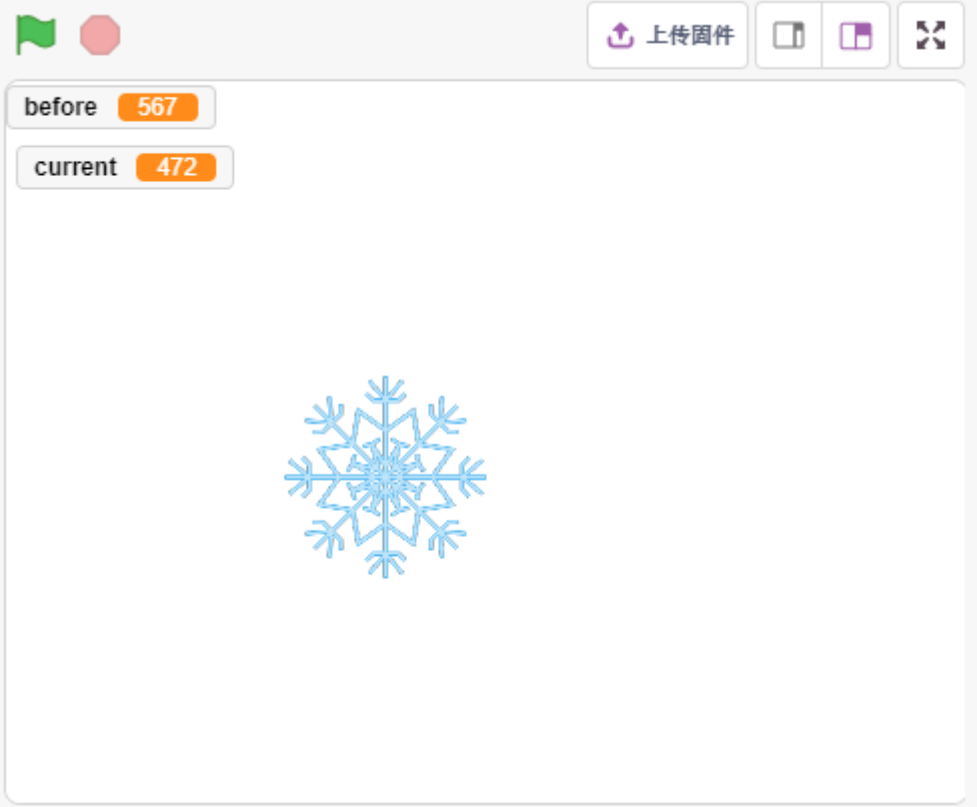
如果读到 pin12 的值为 0，即开关是倾斜的，则将精灵造型切换为 tumbler2, 也是处于倾斜状态。否则，切换精灵造型为 tumbler1, 直立状态。

- [如果 () 那么 () 否则 ()]：如果条件为真，则第一个 C（空格）内的代码将被激活；如果条件为假，则第二个 C 中的代码将激活。

- [=]: 比较运算符, 用来对比等号 2 边的值是否相等，来自运算调色板.

8.3.9 9. 低温报警器

在这个项目中，我们将制作低温报警系统，当温度超过一定时候，舞台上的就会出现雪花标志。



你将学习

- 了解热敏电阻
- 多变量及减法运算

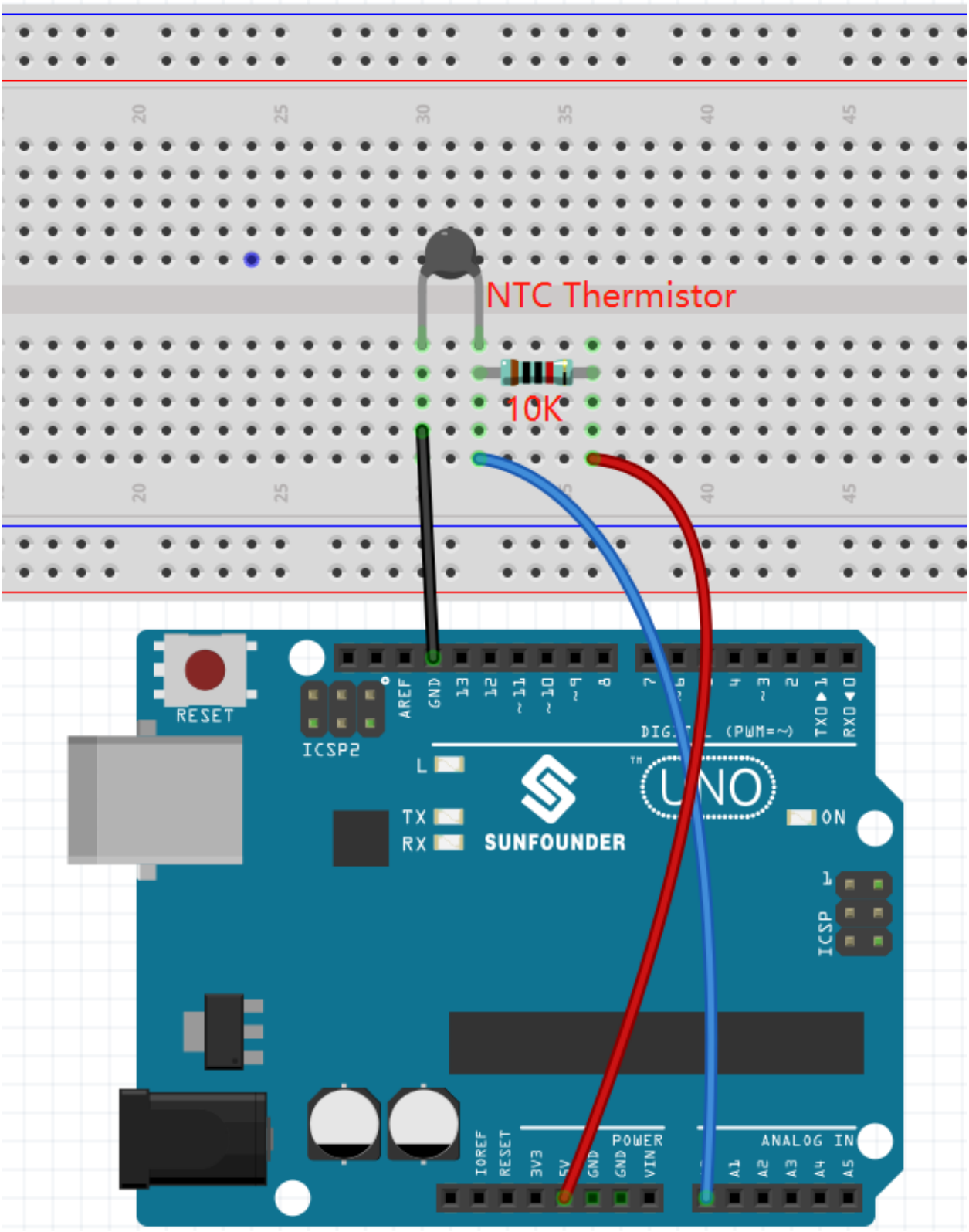
搭建电路

热敏电阻是一种电阻强烈依赖于温度的电阻器，比标准电阻器的电阻更大，并且有两种类型的电阻器，PTC（电阻随温度升高而增加）和 PTC（电阻随温度升高而降低）。

根据下图搭建电路：

热敏电阻一端接 GND，另一端接 A0，串联一个 10K 电阻到 5V。

这里使用的是 NTC 热敏电阻，所以当温度升高时，热 敏电阻的阻值减小，A0 的分压减小，从 A0 得到的值减小，反之增大。


NTC Thermistor
10K
RESET
ICSP2
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
1
0
TXD
RXD
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A1
A2
A3
A4
A5

- 面包板
- 热敏电阻

- 电阻

编程

1. 选择精灵

删除默认精灵，点击精灵区域右下角的 **选择一个角色**按钮，在搜索框中输入 **Snowflake**，然后点击添加。

2. 创建 2 个变量

创建 2 个变量，**before** 和 **current**，用来存放程序运行一开始 A0（热敏电阻）的值以及当前的值。

3. 读取 A0 的值

当绿色旗子点击时，读取 A0 的值并存放到变量 **before**。

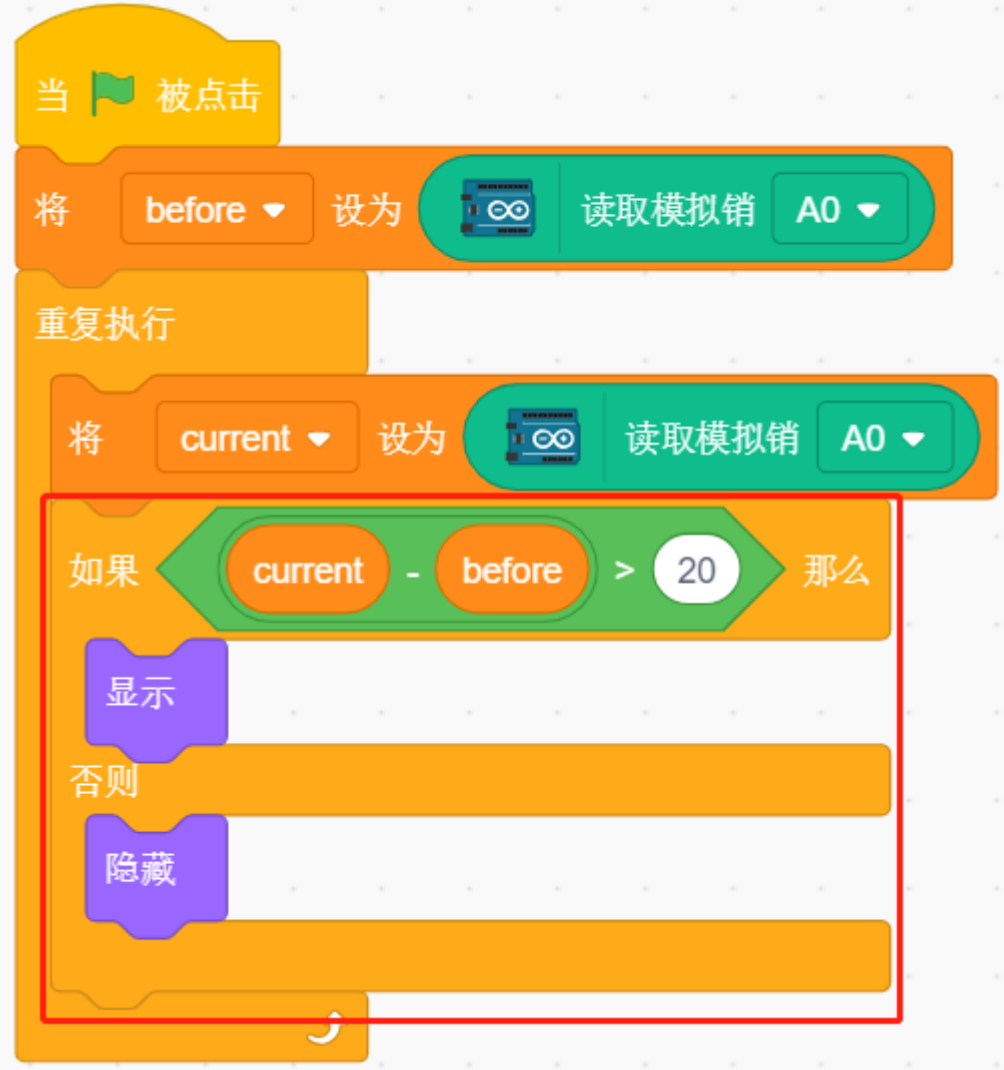
4. 再次读取 A0 的值

在 [重复执行] 中，再次读取 A0 的值存放到变量 **current**。

5. 判断温度变化

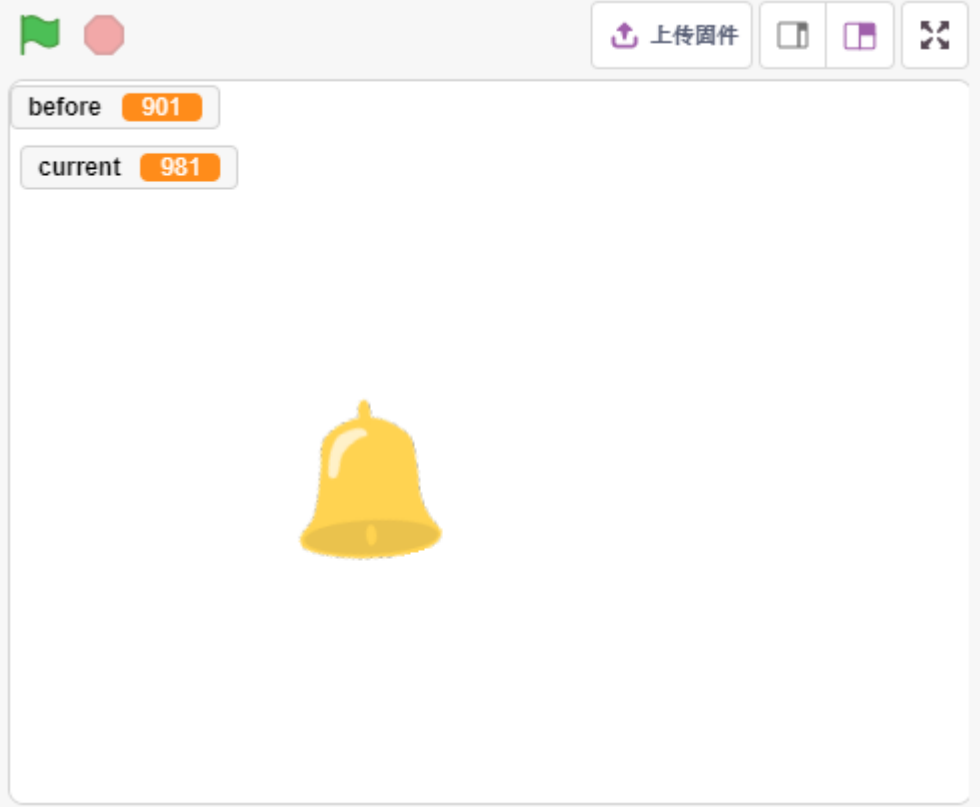
使用 [如果 <> 那么 <> 否则 <>] 块, 判断当前的 A0 值是否比之前大 50，即温度降低了。此时让 Snowflake 精灵显示，否则隐藏.

- [如果 <> 那么 <> 否则 <>]: 条件判断块来自 **控制**调色板。判断 **如果**的条件是否满足，如果满足，运行里面的块; 否则运行 **否则**内的块.
- [-] & [>]: 减法运算符和比较运算符，来自 **运算**调色板。
- [显示] & [隐藏]: 显示和隐藏精灵。



8.3.10 10. 光控闹钟

生活中，有各种各样的时间闹钟。现在我们来制作一个光控闹钟，早晨来临，光亮度增强，这个光控闹钟就会提醒你，该起床了。



你将学习

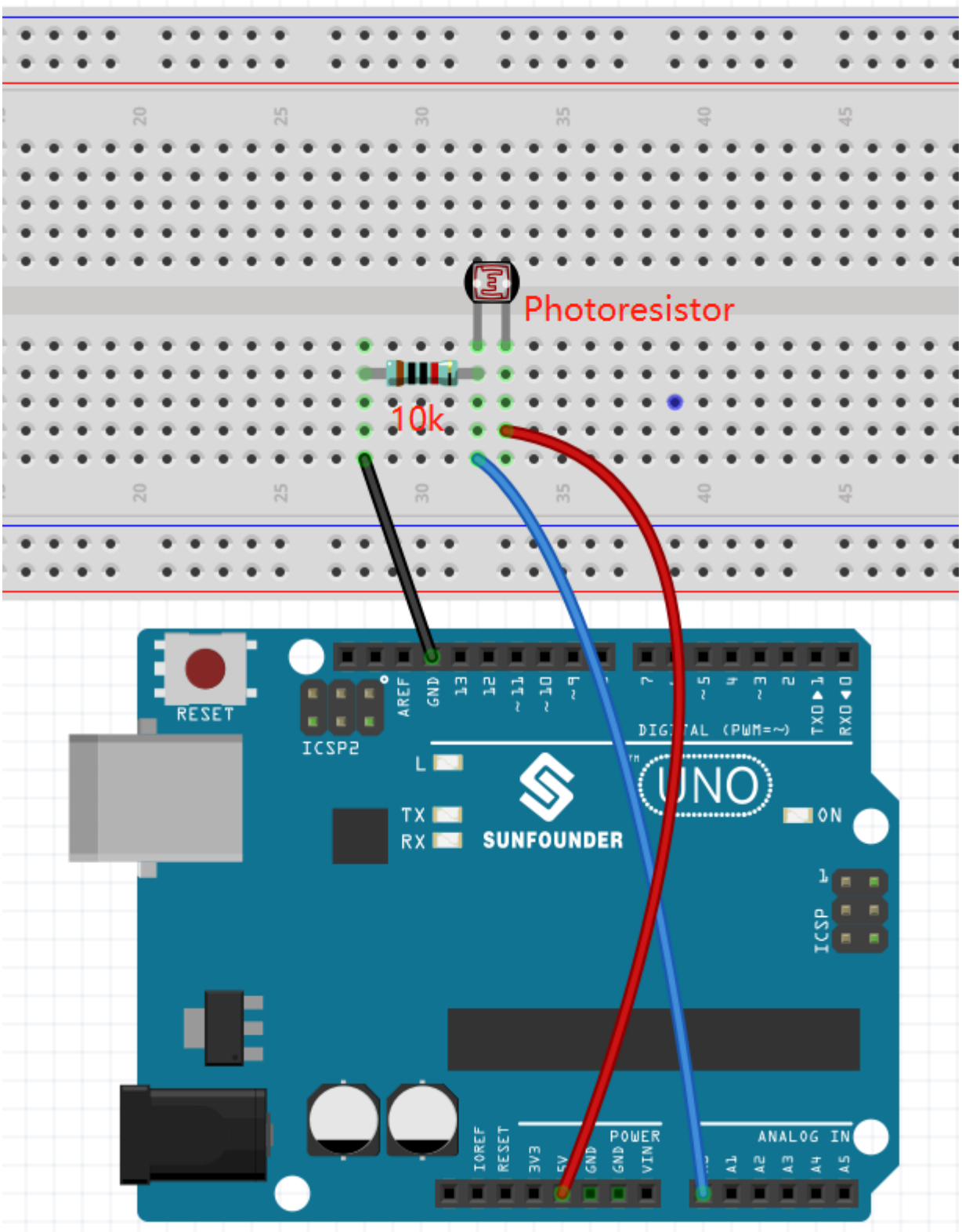
- 了解光敏电阻
- 停止播放声音和停止脚本运行

搭建电路

光敏电阻或光电管是一种光控可变电阻器。光敏电阻的电阻随着入射光强度的增加而降低。

根据下图搭建电路：

光敏电阻一端接 5V，另一端接 A0，在此端与 GND 串联一个 10K 的电阻。所以当光照强度增加时，光敏电阻的阻值减小，10K 电阻的分压增加，A0 得到的值变大。


Photoresistor
10k
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A1
A2
A3
A4
A5
ANALOG IN

- 面包板
- 光敏电阻
- 电阻

编程

1. 选择精灵

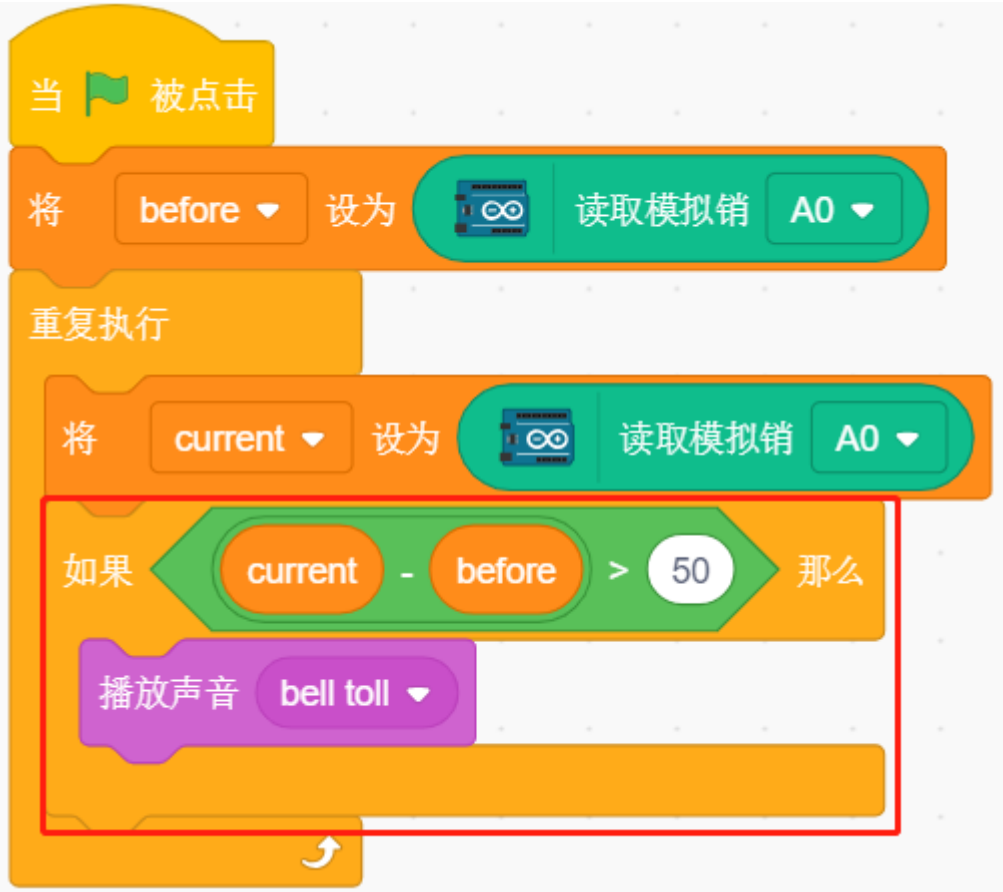
删除默认精灵，点击精灵区域右下角的 **选择一个角色**按钮，在搜索框中输入 **bell**，然后点击添加。

2. 读取 A0 的值

创建 2 个变量 **before** 和 **current**, 程序一开始，读取 A0 的值存放到 before 中以作为参考值。在 [重复执行] 中，再次读取 A0 的值，存放到变量 current 中，重复循环的读取。

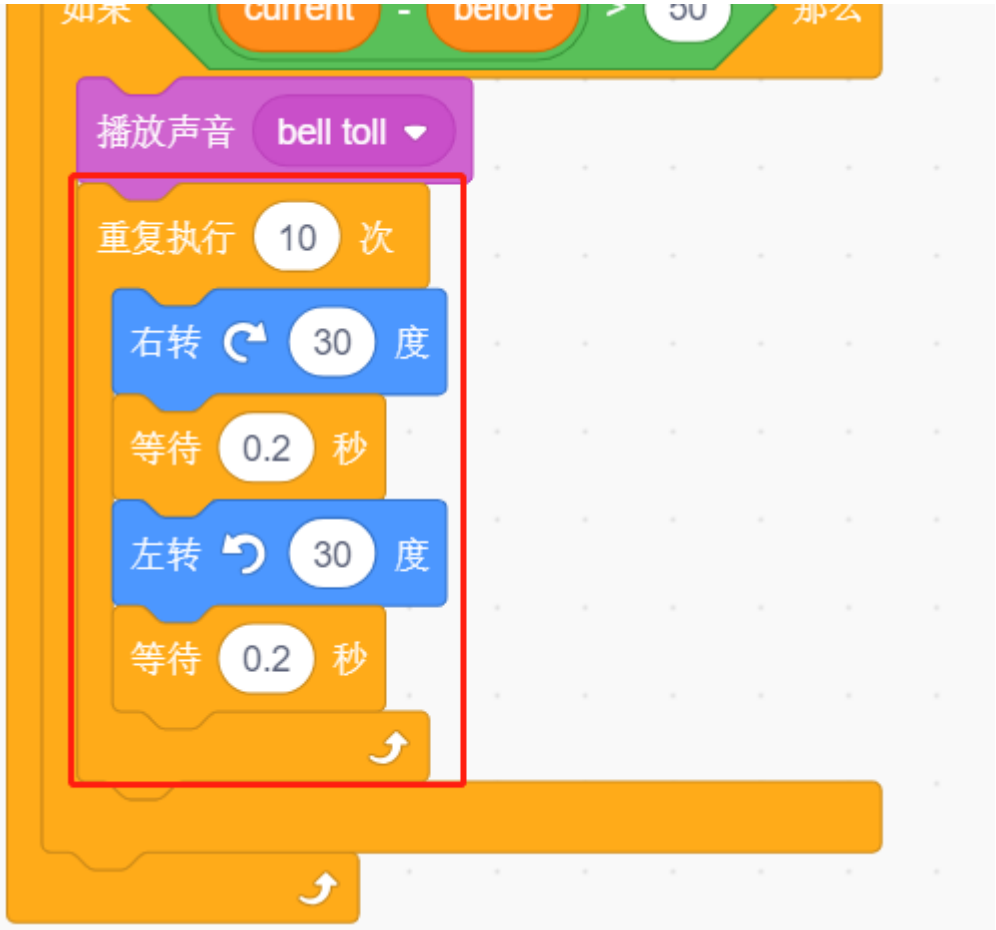
3. 发出声音

当 **current** 的值比 **before** 大 50，即当前光强度大于设定值，则让精灵发出声音。



4. 转动精灵

使用 [右转/左转（）度] 来让 bell 精灵左右转动来达到闹铃的效果。

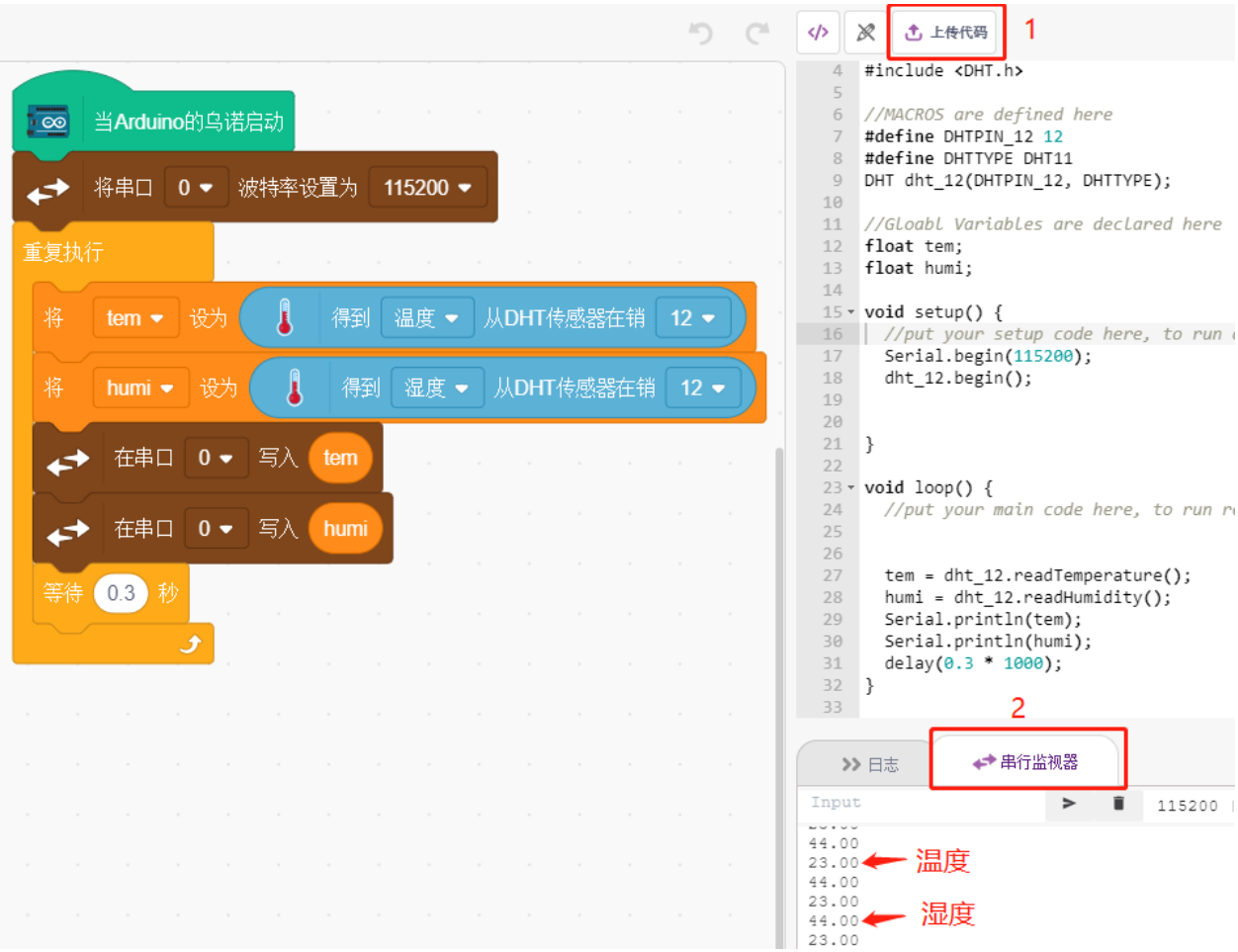


5. 停止所有

当闹钟响了一段时间后，停止运行。

8.3.11 11. 读取温湿度

之前的项目一直使用舞台模式，但是有些功能只有上传模式才有，比如串口通讯功能。在这个项目中，我们将在上传模式 下使用串行监视器打印 DHT11 的温度和湿度。



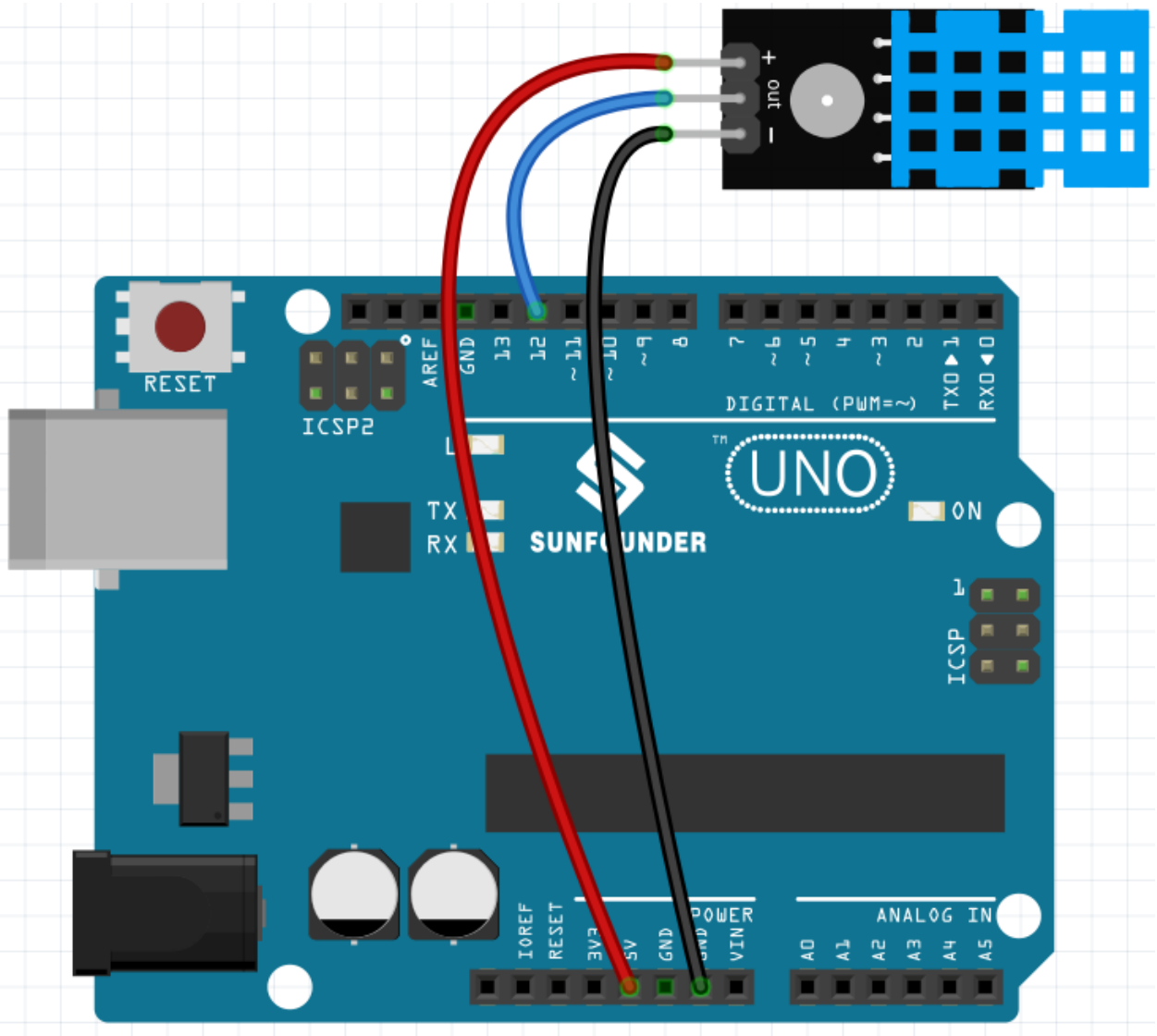
You Will Learn

- 通过 DHT11 模块得到温湿度
- 上传模式的串行监视器
- 添加扩展

搭建电路

数字温湿度传感器 DHT11 是一种复合传感器，包含经过校准的温湿度数字信号输出。

现在根据下图构建电路:



- 面包板
- 光敏电阻
- 电阻

编程

1. 添加拓展

切换到上传模块，点击左下角的 **添加拓展**按钮。

然后选择 **通讯**来将它添加，最后它将出现在调色板区域的最后面。

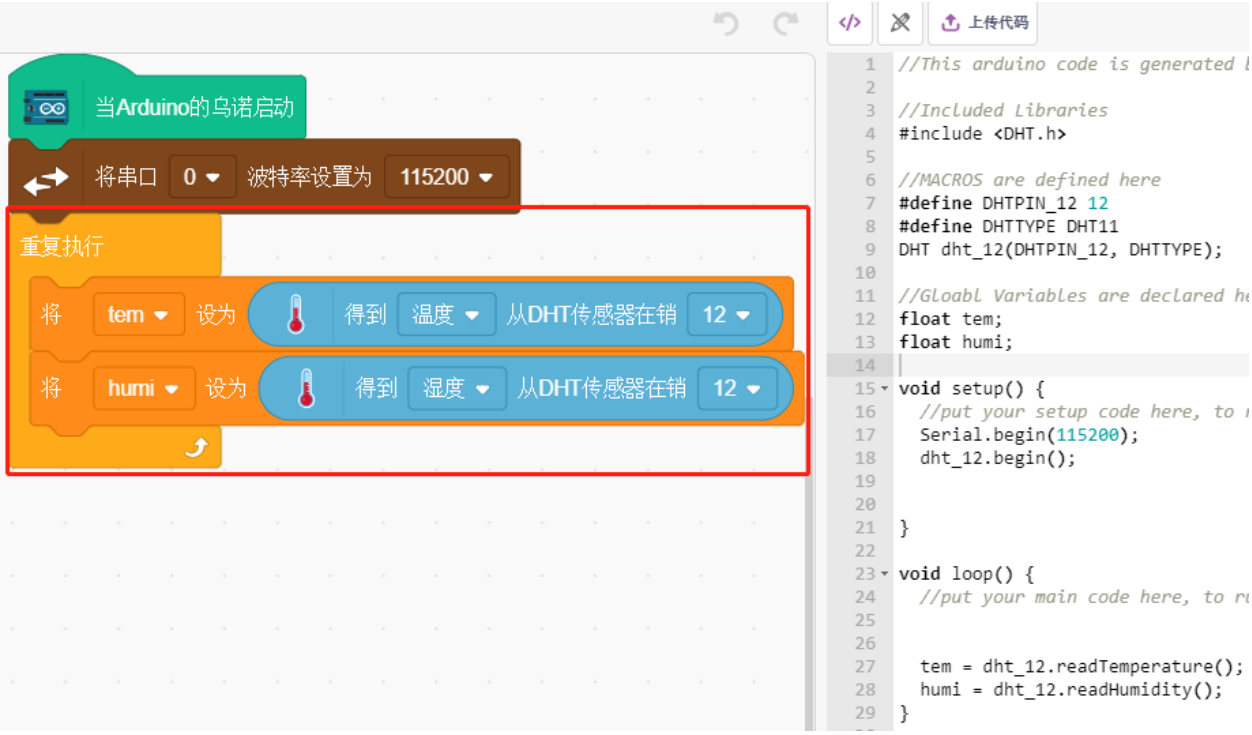
2. 初始化 Arduino 的乌诺和串口监视器

在上传模式中，启动 Arduino 的乌诺，然后设置串口波特率。

- [当 Arduino 的乌诺启动]: 在上传模式中，启动 Arduino 的乌诺。
- [将串口 () 波特率设置位 ()]: 来自通讯调色板, 用来为串口 0 设置波特率，默认为 115200. 如果你用的是 Mega2560, 那你可以选择在串口 0~3.

3. 创建变量并读取温湿度

创建 2 个变量 **tem** 和 **humi** 用来分别存放温度和湿度，在你拖拽块的同时，右侧会出现相关的代码。



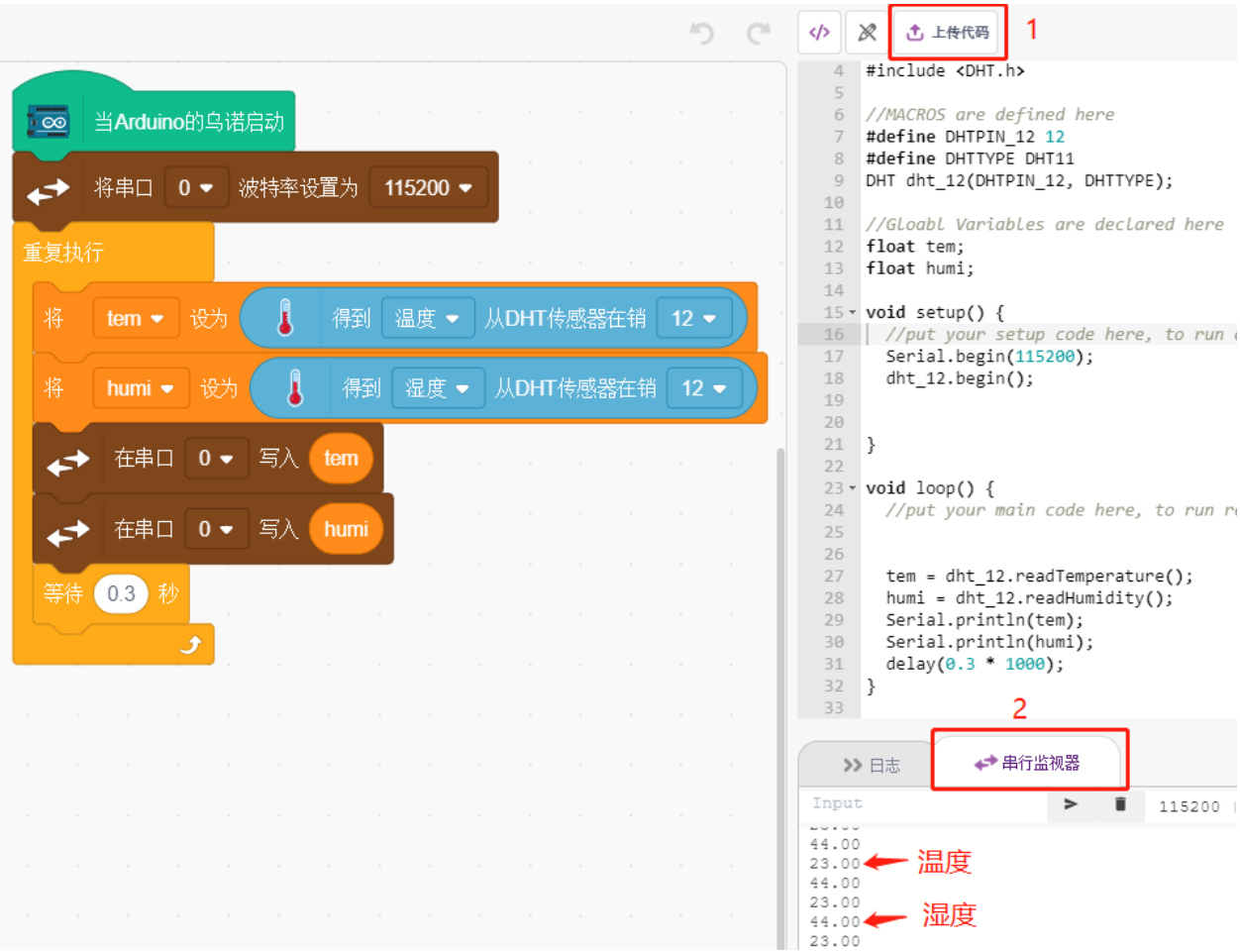
4. 在串口打印温湿度

将读取到的温湿度写到串口中，为避免传输速度太快，导致 PictoBlox 卡机，用 [等待 () 秒] 块, 为下一次打印增加一些时间间隔。

5. 上传代码并打开串口监视器

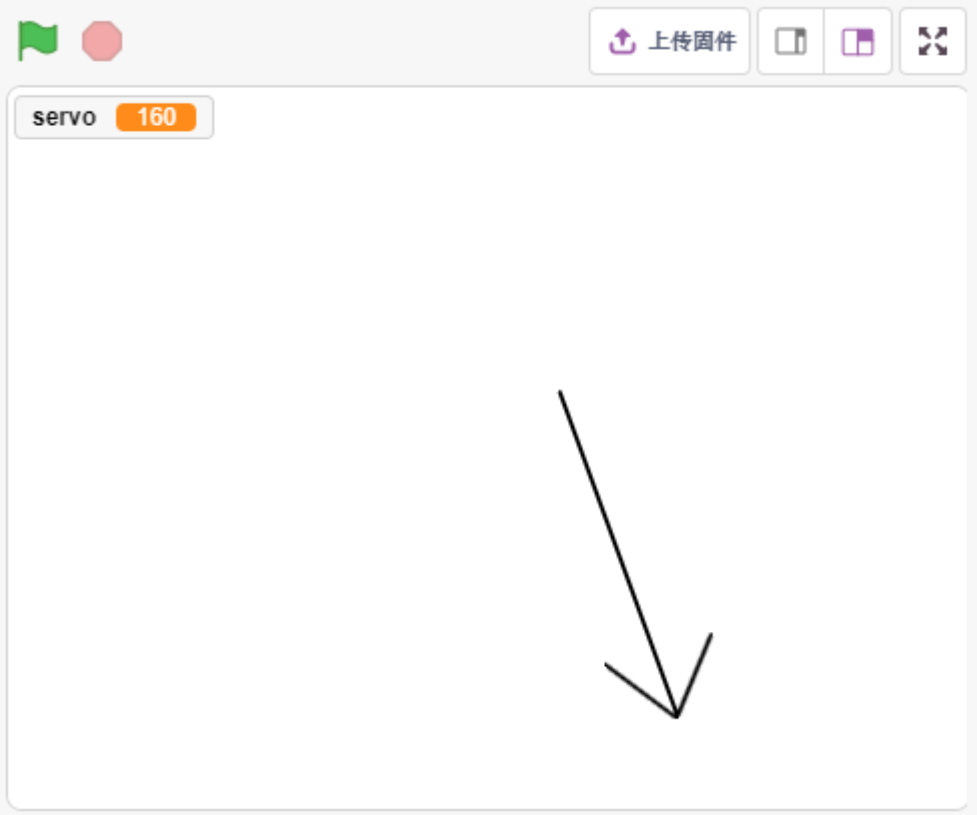
与舞台模式不同的是，上传模式的代码需要用 **上传代码**按钮上传到 Arduino 板子上才能看到效果，这也让你可以在拔掉 USB 线之后，程序仍然在运行。

现在打开串口监视器来查看温湿度。



8.3.12 12. 摆钟

在这个项目中，我们将制作一个箭头钟摆，同时舵机会跟着转动。



你将学习

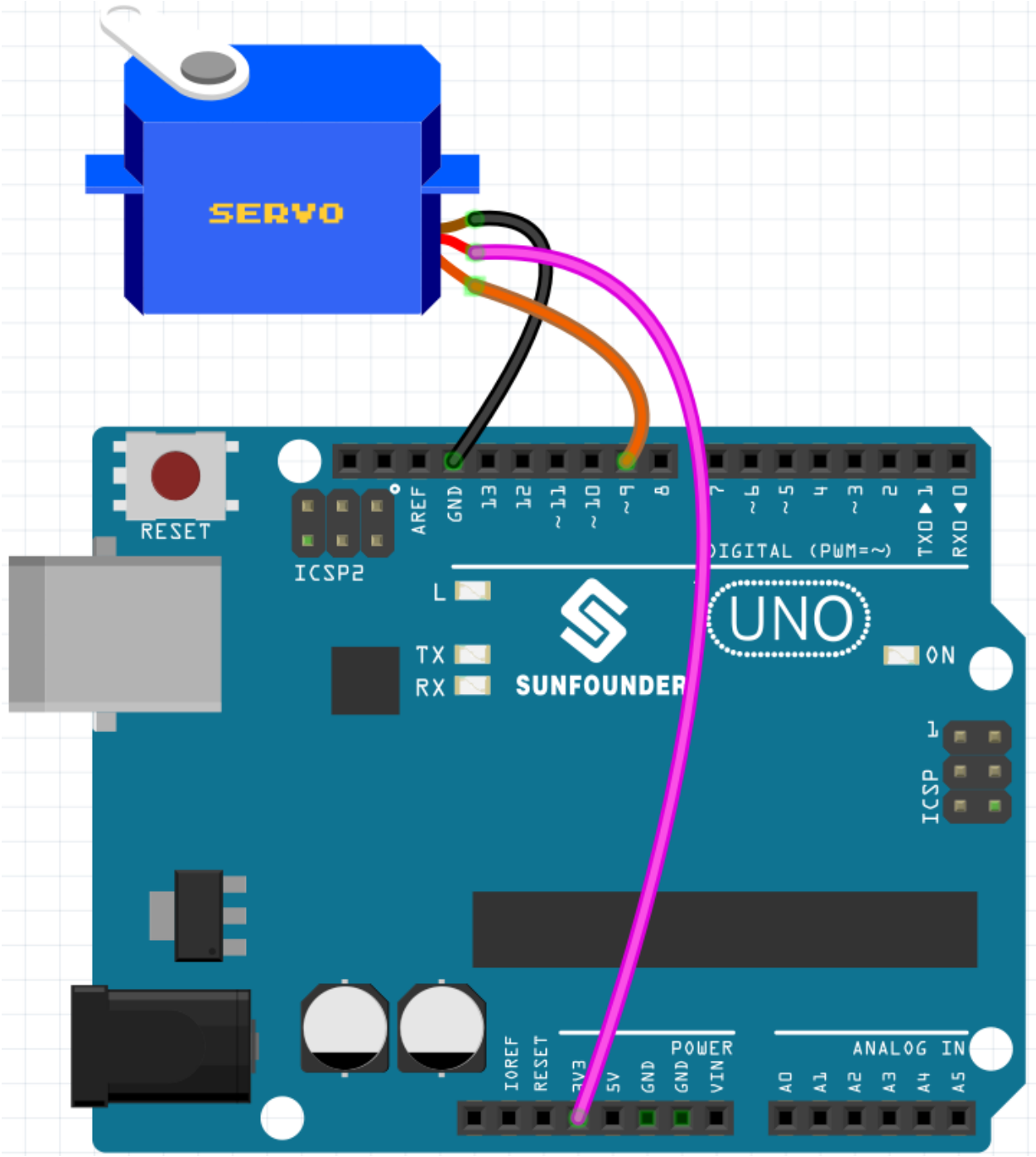
- 舵机的工作原理及角度范围
- 画一个精灵，将中心点放尾部。

搭建电路

伺服是一种只能旋转 180 度的齿轮电机。它是通过从电路板发送电脉冲来控制的。这些脉冲告诉伺服它应该移动到什么位置。

舵机共有三根线：棕色线为 GND，红色线为 VCC（接 3.3V），橙色线为信号线。角度范围为 0-180。

现在根据下图构建电路：


SERVO
RESET
ICSP2
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TXD 1
RXD 0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

- 面包板
- 舵机

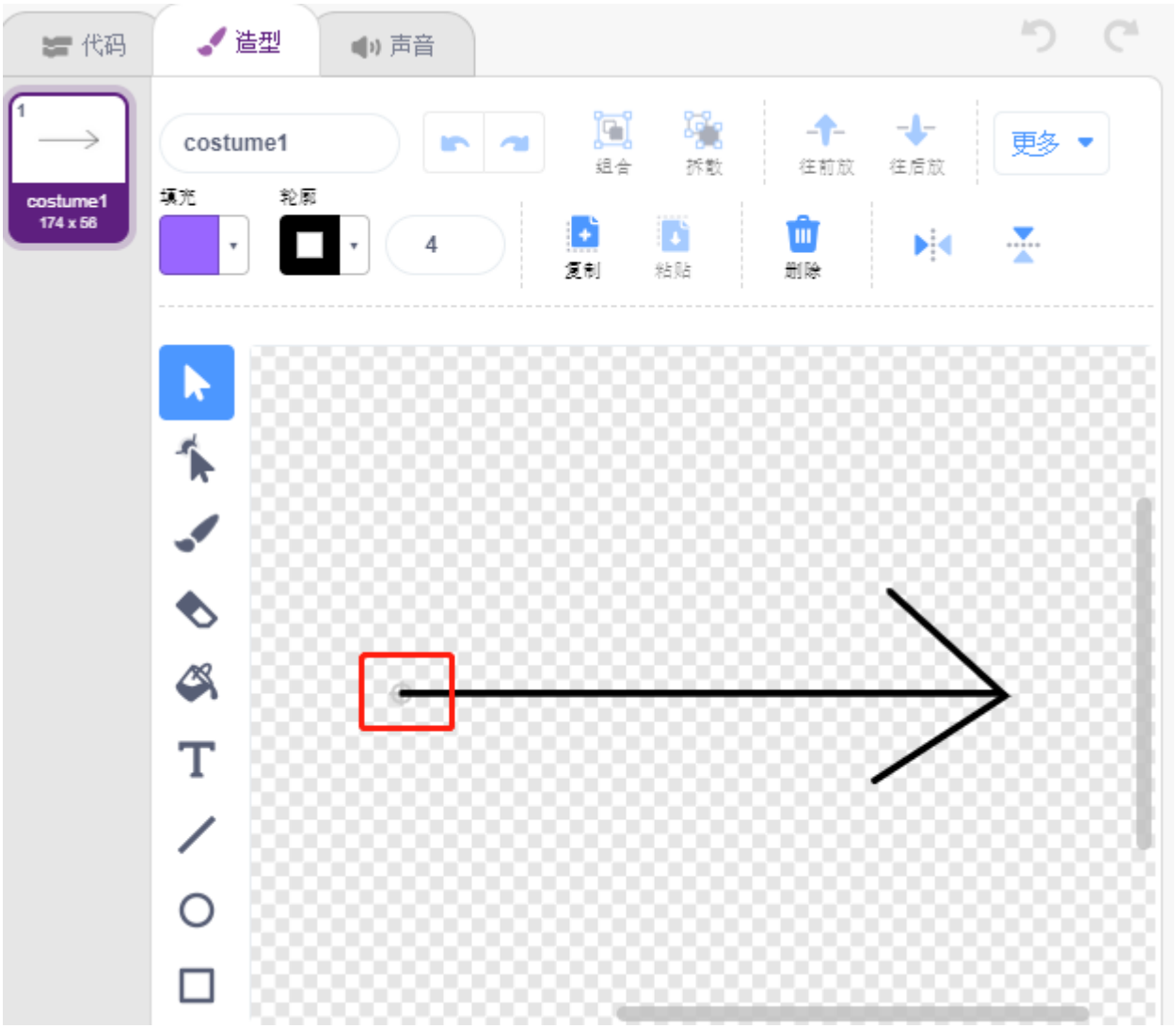
编程

1. 画一个精灵

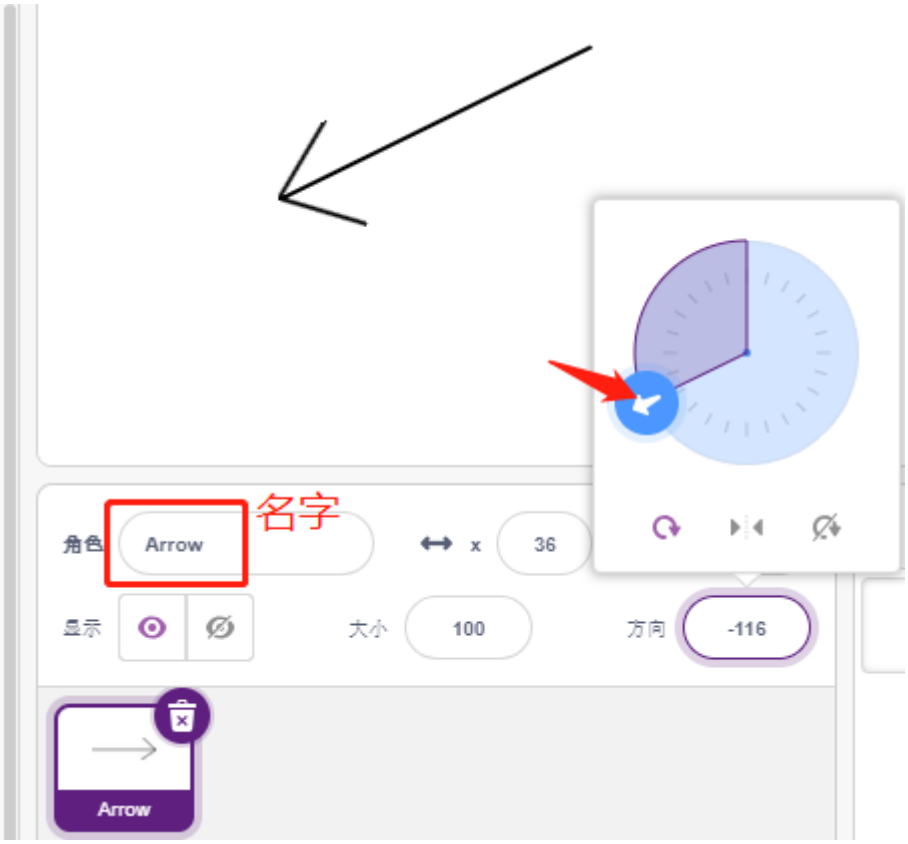
删除默认的精灵，选择精灵按钮，点击 **绘制**，将出现一个空白精灵 Sprite1。

在打开的 **造型**页面，使用 **画线段**工具画一个箭头。

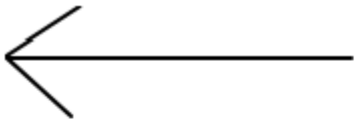
备注： 一定要从画布的中心点向外开始画箭头，这样就能保证箭头是以中心点为原点转圈。按住 Shift 可让线角度为直的或 45 度方向。

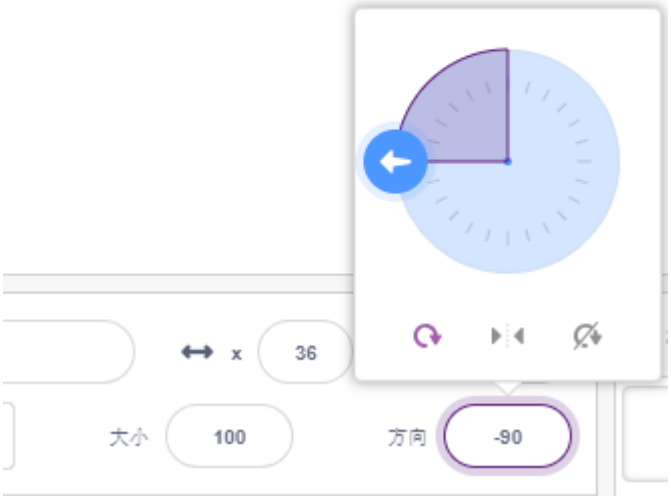


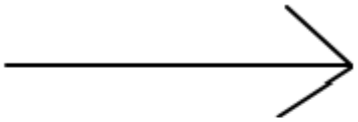
画完之后，将在舞台上显示箭头精灵，将它命名为 **arrow**。然后点击 **方向**后的 270，将出现一个圆形表盘，现在拖动这个箭头，看下舞台上的 arrow 精灵是否以尾部为原点转圈。

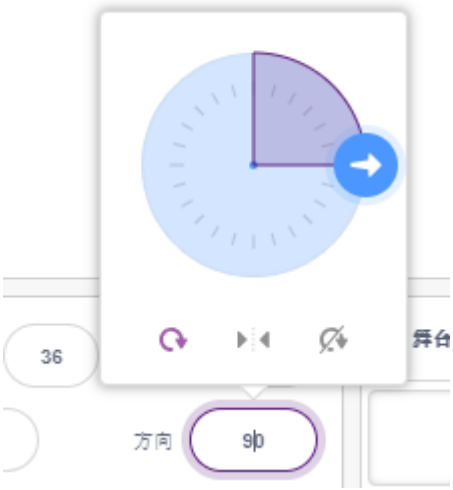


若要让 arrow 精灵从左边摆动到右边，角度范围是-90~-180，180~90。









2. 创建变量

创建一个变量 servo, 用来存放角度值，并设置初始值为 270。

3. 从左边摆动到右边

现在让箭头精灵从左边-90 度位置摆动到右边 90 度位置。

通过 [重新 () 次] 块, 将变量每次加-10，通过 18 次就能加到 90 度。然后用 [面向 () 方向] 让 arrow 精灵转到这些角度。

由于精灵转动角度是-180 ~ 180，超出这个范围的角度会通过下面的条件来换算。

- 如果角度 > 180，则角度-360.

4. 转动舵机

当你点击绿色旗子的时候，你会发现箭头迅速转动到右边，然后回到左边，所以这里用一个 [等待 () 秒] 块让转动速度慢点。另外使用 [上 () 组伺服至 () 角] 块让接在 Arduino 板上的舵机转到特定的角度。

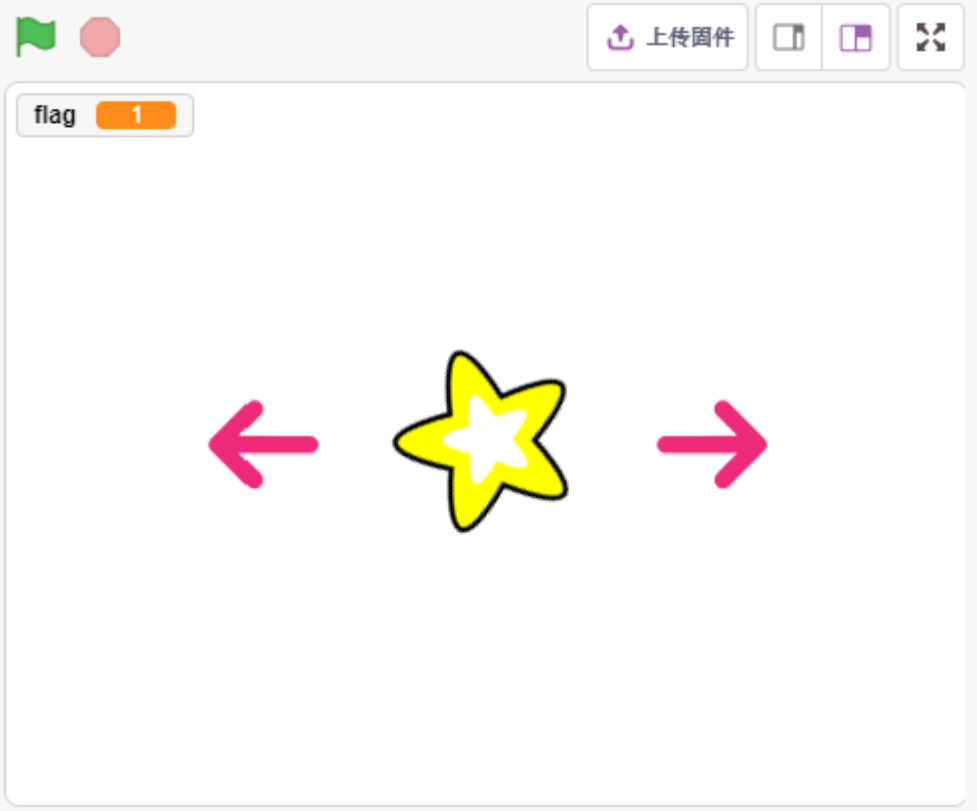
5. 从右边摆动到左边

通过同样的方法，让舵机和箭头精灵从右边慢慢转动到左边。

8.3.13 13. 旋转风扇

在这个项目中，我们将制作一个旋转的星星精灵和风扇。

点击舞台上的左右箭头精灵将控制电机的正转和反转，同时星星精灵也会随着转，点击星星精灵时，点击停止转动。



你将学习

- 电机工作原理
- 广播功能
- 停止精灵块中的其他脚本

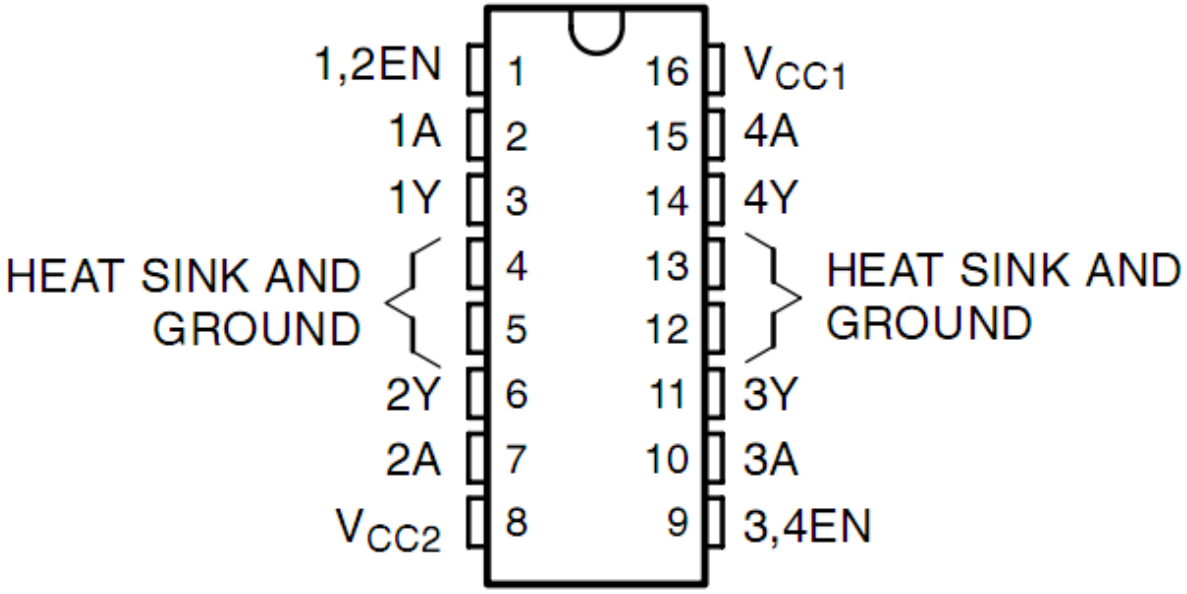
搭建电路

在本项目中，电机驱动芯片*L293D* 用于使电机旋转。

L293D 是一款高电压大电流芯片集成的 4 通道电机驱动器。

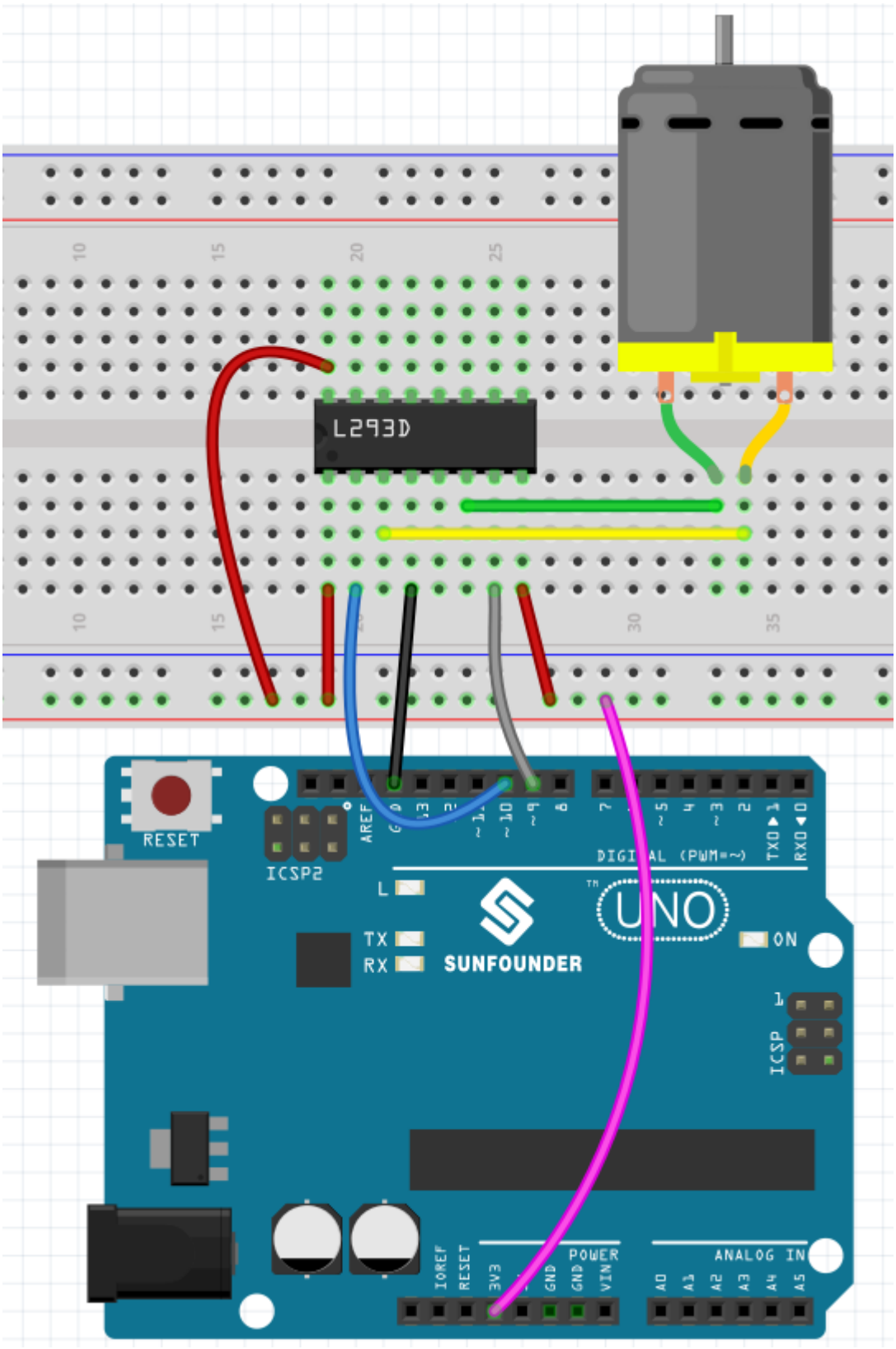
其引脚排列如下所示：

EN 引脚为使能引脚，只工作在高电平；A 代表输入，Y 代表输出。当销 EN 是高电平时，如果甲是高，ÿ 输出高电平; 如果 A 为低电平，则 Y 输出低电平。当 EN 引脚为低电平时，L293D 不工作。



现在根据下图构建电路:

- L293D 的 Enable pin 1,2EN 已经连接到 3.3V，所以 L293D 一直处于工作状态。
- 将引脚 1A 和 2A 分别连接到控制板的引脚 9 和 10。
- 电机的两个引脚分别连接到引脚 1Y 和 2Y。
- 当 10 脚为高电平，9 脚为低电平时，电机开始向一个方向旋转。
- 当引脚 10 为低电平且引脚 9 为高电平时，它以相反的方向旋转。


L293D
RESET
ICSP2
AREF
DIGITAL (PWM=~)
TXD
RXD
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
GND
GND
VIN
A0
A1
A2
A3
A4
A5

- 面包板
- 直流电机

- *L293D*

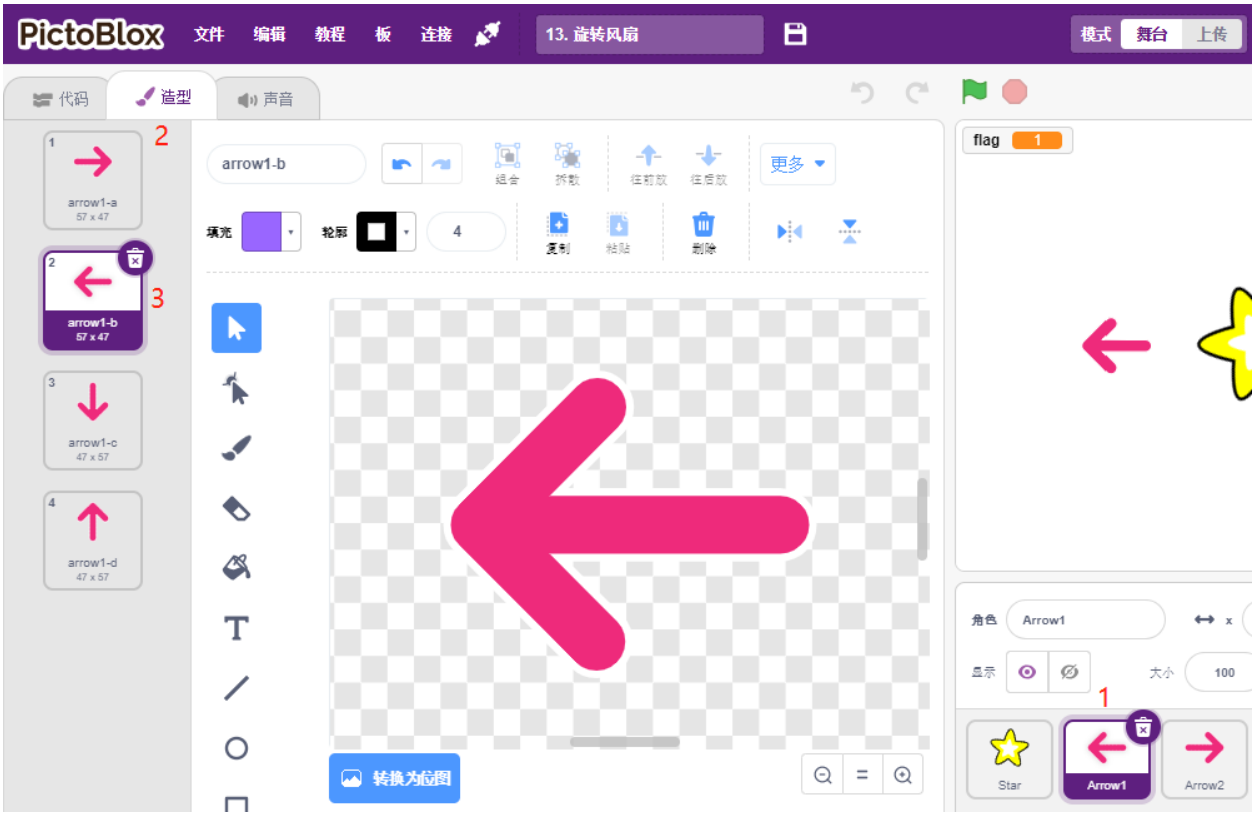
编程

我们要实现的效果是用 2 个箭头精灵分别控制电机和 **Star** 精灵的正转和反转，点击 **Star** 精灵，电机将停止转动。

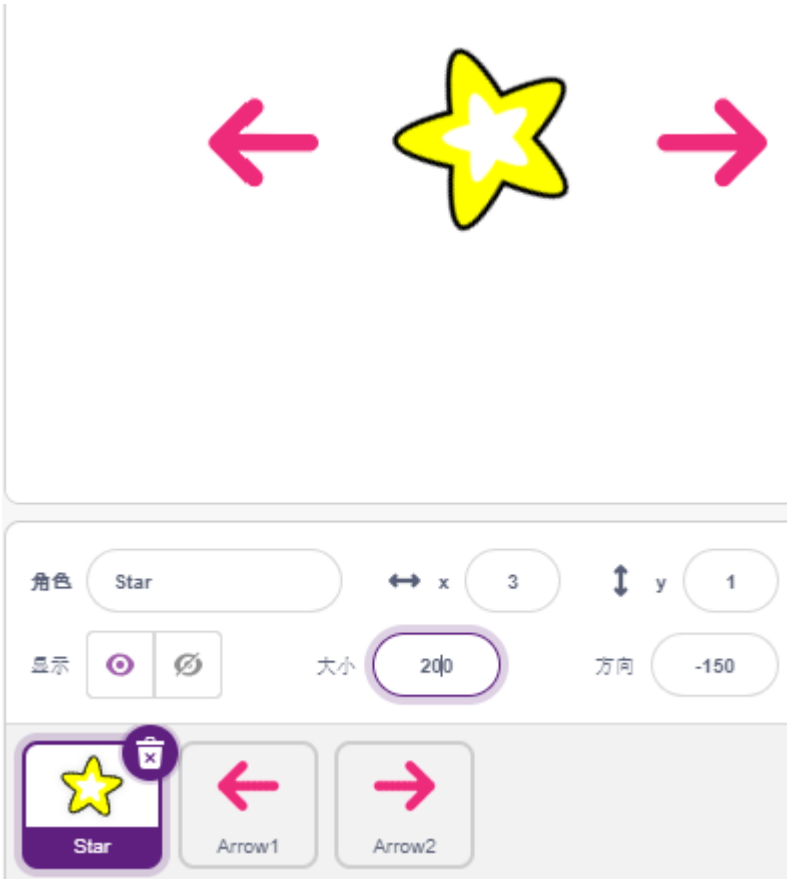
1. 添加精灵

删除默认精灵，然后选择 **Star** 精灵和 **Arrow1** 精灵，并复制 **Arrow1** 一次。

在 **造型**页面中，将 **Arrow1** 精灵更改为不同方向的造型。



适当调整精灵的大小和位置。



2. Arrow1 精灵

当此精灵被点击时，广播一条信息 turn, 然后设置数字引脚 9 为低电平和 pin 10 为高电平，将变量 flag 设置为 1. 若你点击 **Arrow1** 精灵，你会发现电机逆时针转，如果你的顺时针转，那你交换下 9 引脚和 10 引脚的位置。

这里有 2 个点需要注意：

- [广播 ()]: 来自事件调色板，用于广播一条信息给其他精灵，当其他精灵收到这条信息，将执行特定的事件。比如这里是 turn，当 **Star** 精灵接收到这个信息，它就执行转动的脚本。
- 变量 [flag]：**Star** 精灵转动方向是通过 flag 的值来判定的，flag 为 1，让 **Star** 精灵反转; flag 为 0，则正转。所以你在创建 flag 变量的时候，需要让它适用于所有的精灵。

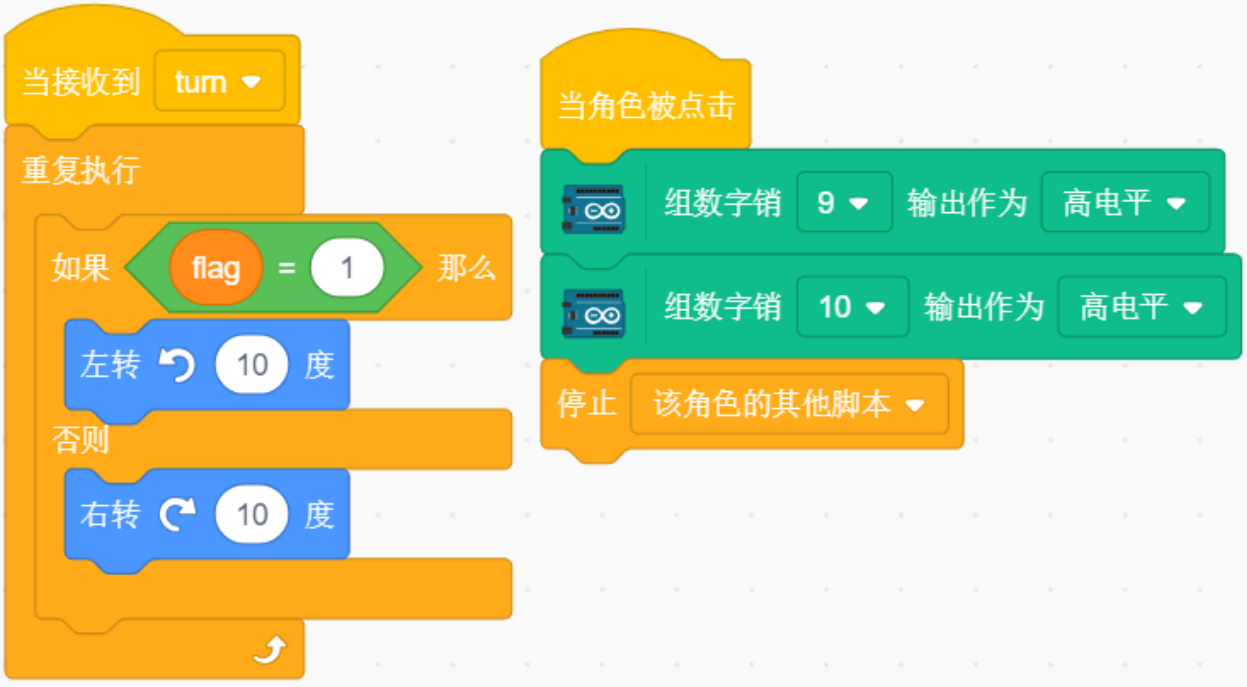
3. Arrow2 精灵

当此精灵被点击时，广播一条信息 turn, 然后设置数字引脚 9 为高电平和引脚 10 为低电平来让电机顺时针转动，并将变量 flag 设置为 0.

4. Star 精灵

这里包含 2 个事件：

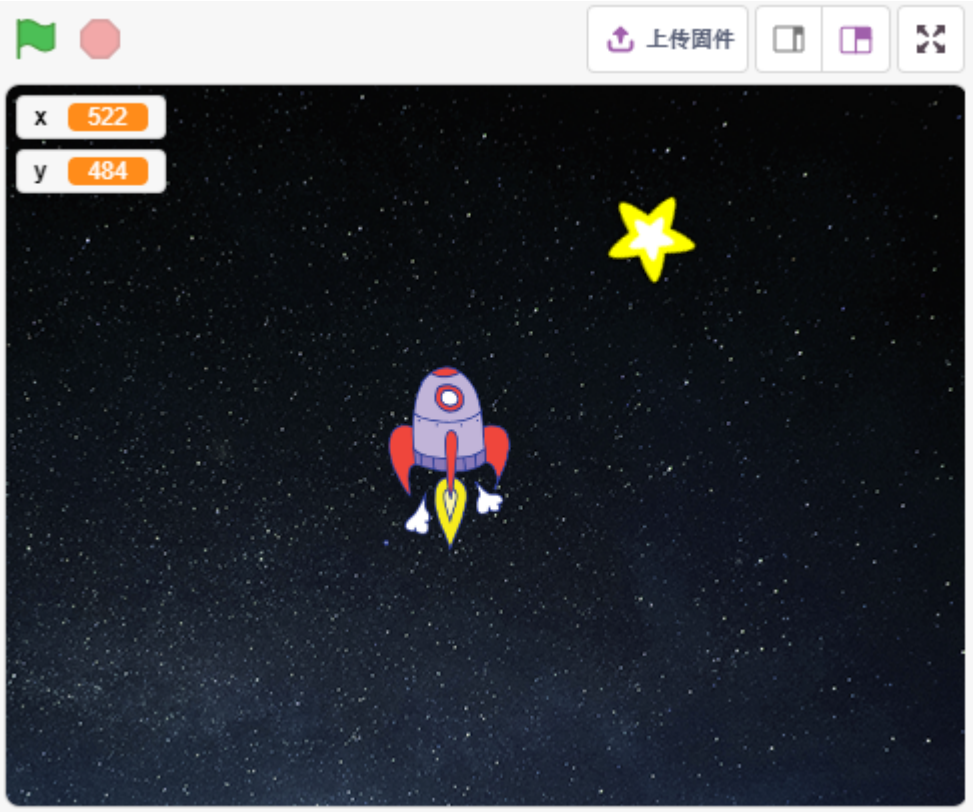
- 当 **Star** 精灵接收到广播的信息 turn 时，则判断 flag 的值，如果 flag 为 1，向左转 10 度，否则反转。由于是在 [重新执行] 中，所以它会一直转动。
- 当这个精灵被点击，将电机的 2 个引脚都设置为高电平来让它停止转动，并停止在这个精灵中的其他脚本。



8.3.14 14. 游戏 - 星际穿越

在这里，我们使用摇杆模块来玩一个躲避星星的游戏：

脚本运行后，舞台上会随机出现星星，你需要用摇杆控制火箭精灵躲避星星，如果碰到了，游戏将结束。



你将学习

- 摇杆模块工作原理
- 根据摇杆的值来设置精灵的 x，y 坐标
- 让精灵去随机的位置
- 精灵之间的触碰事件

搭建电路

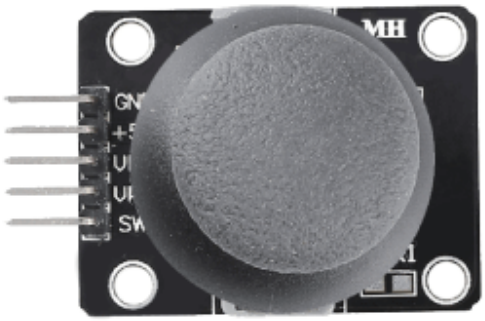
操纵杆是一种输入设备，由一个在底座上旋转的操纵杆组成，并向它所控制的设备报告其角度或方向。操纵杆通常用于控制视频游戏和机器人。

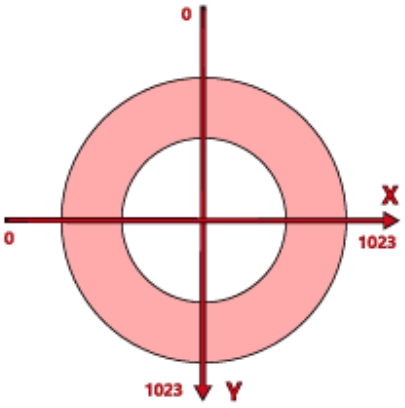
为了将完整的运动范围传达给计算机，操纵杆需要在两个轴上测量操纵杆的位置——X 轴（从左到右）和 Y 轴（上下）。

摇杆的运动坐标如下图所示：

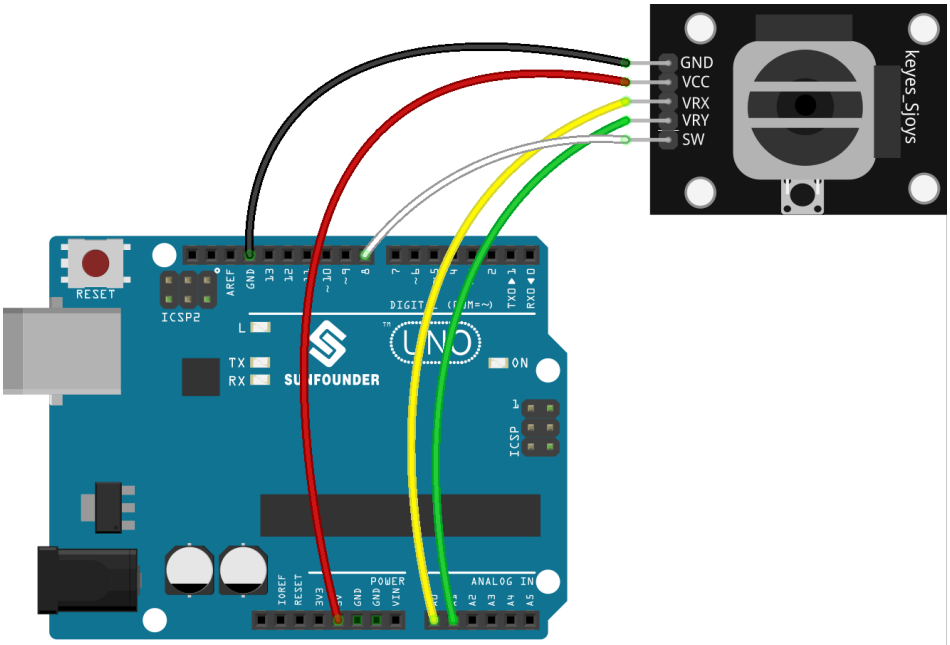
备注：

- x 坐标是从左到有，范围是 0-1023
- y 坐标是从下到上，范围是 0-1023





现在按照下图搭建电路.



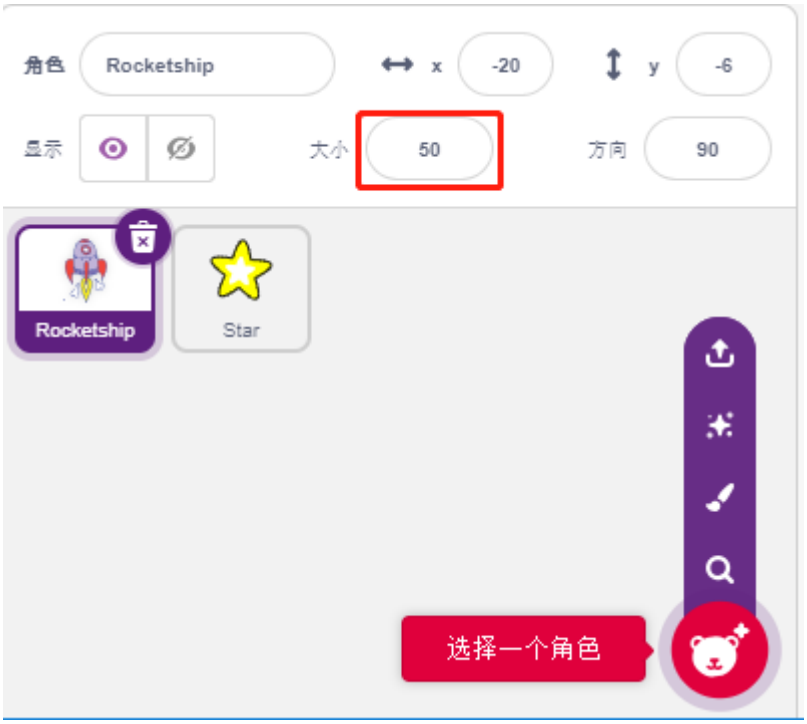
- 面包板
- 摇杆模块

编程

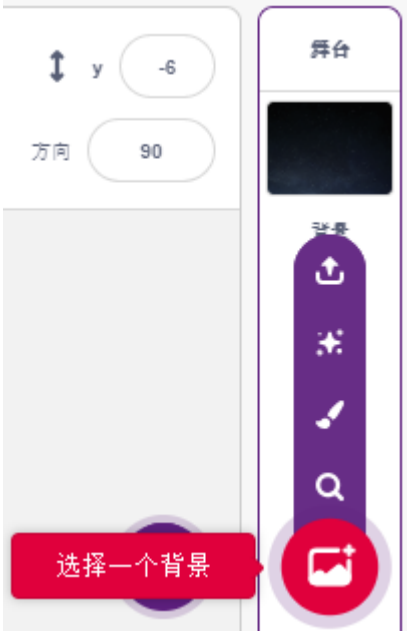
整个脚本要实现的效果是，当绿旗点击时，Star 精灵在舞台上呈曲线运动，你需要用摇杆来移动 Rocketship，以免它被 Star 精灵碰到。

1. 添加精灵和背景

删除模块精灵，用 **选择一个角色**按钮来添加 **Rocketship** 精灵和 **Star** 精灵。注意将 Rocket 的尺寸设置为 50%。



现在通过 **选择一个背景**来添加 **Stars** 背景。



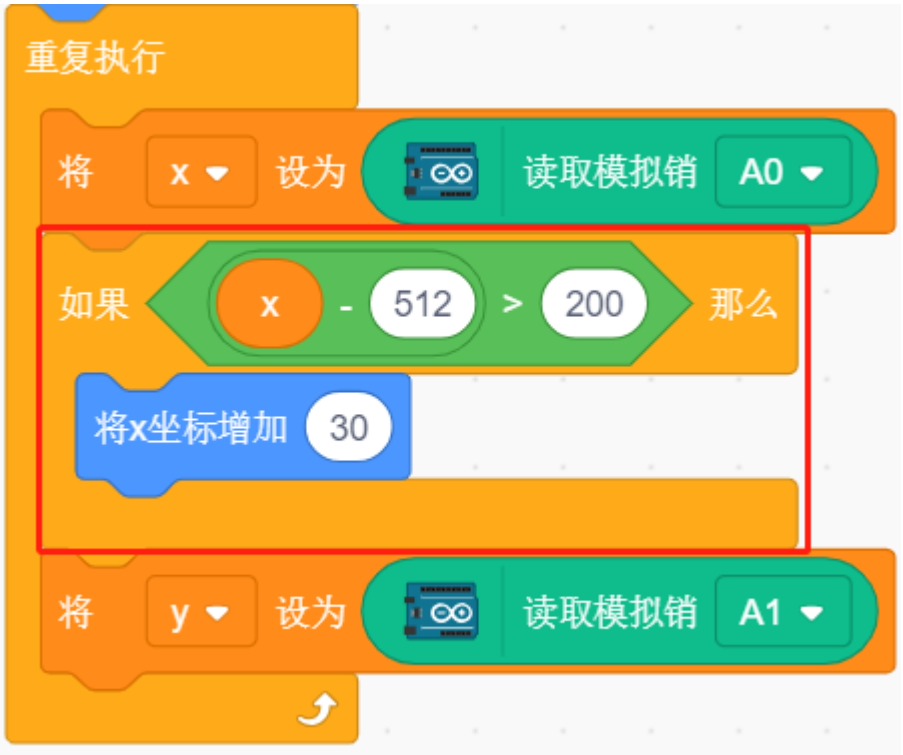
2. 为 Rocketship 编写脚本

Rocketship 精灵要实现的效果，它会在随机位置出现，然后由摇杆控制它的上下左右移动。

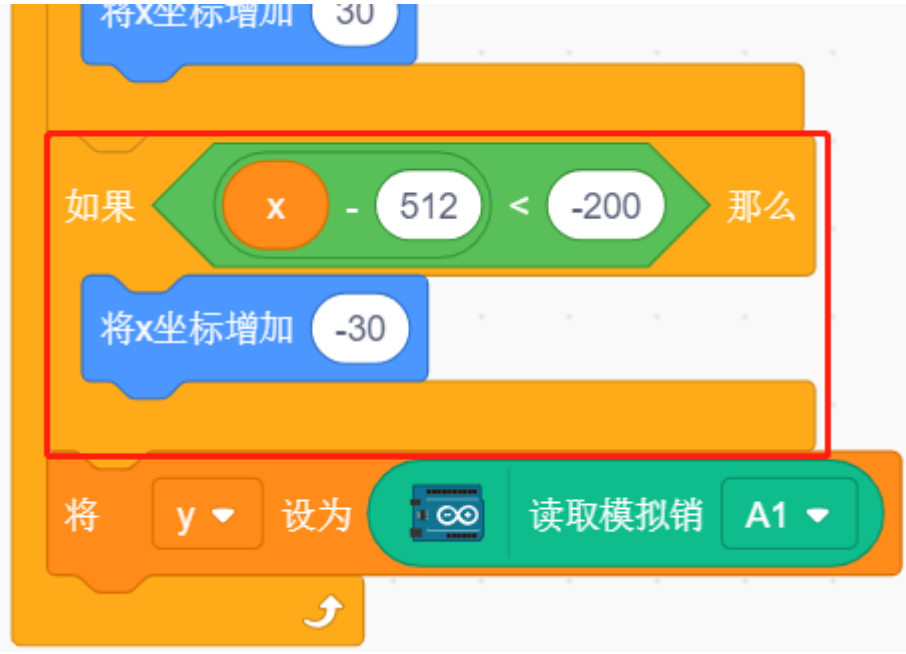
工作流程如下：

- 当绿旗被点击时，让精灵去随机的位置，创建 2 个变量 x 和 y, 分别存放从 A0（VRX of Joystick）和 A1（VRY of Joystick）的读的值。你可以让脚本运行，上下左右拨动摇杆，看下 x 和 y 的取值范围。

- A0 的值的范围时 0-1023（中间位置约为 512），用 [x-512>200] 来判断 Joystick 是否向右拨动，如果是则让精灵的 x 坐标 +30（让精灵向右移动）。



- 如果向左拨动 [x-512<-200], 则如果是则让精灵的 x 坐标-30（让精灵向左移动）。



- 由于 Joystick 的 y 坐标是从上（0）到下（1023），而精灵的 y 坐标是从下（-211）到上（211）。所以为了能在向上拨动摇杆，精灵也向上移动，须在脚本中让 y 坐标-30。

- 如果摇杆下拨拨动，精灵 y 坐标 +30.

3. 为 Star 编写脚本

Star 精灵要实现的效果是在随机的位置出现，并且碰到 Rocketship, 则停止脚本运行，游戏结束。

- 当绿旗被点击，精灵去随机的位置，[右转（）度] 块是让 Star 精灵向前移动的同时，有点角度变化，这样你就能看到它是呈曲线运动，并在碰到边缘就反弹。

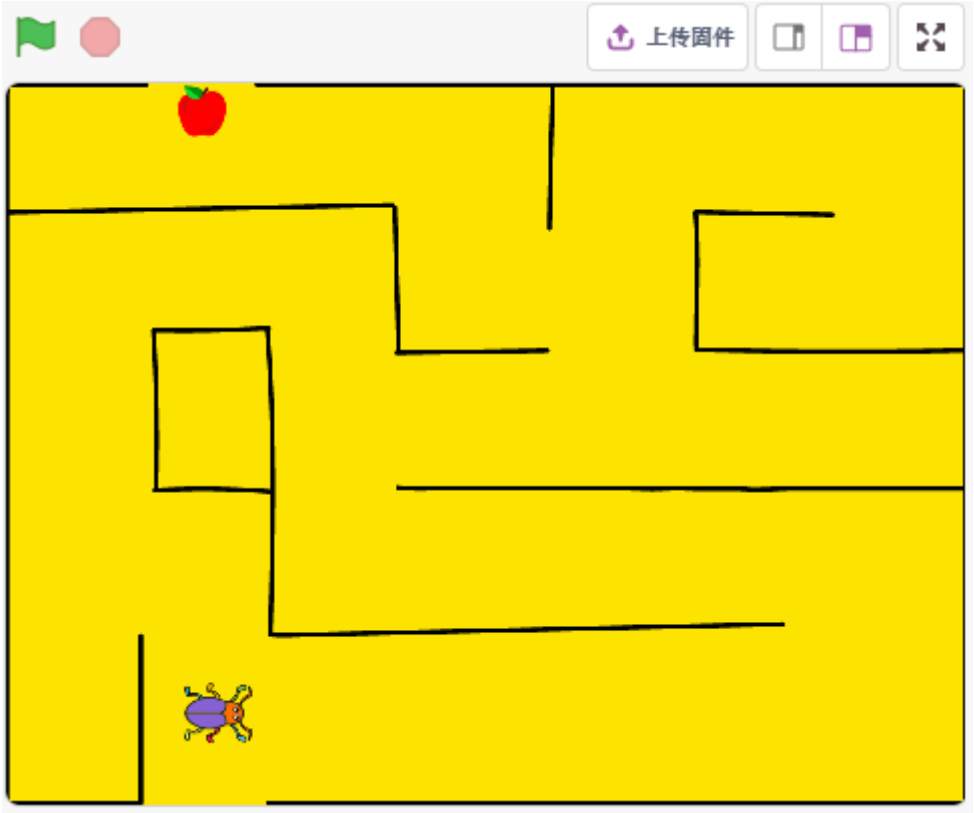
- 如果精灵在移动过程中碰到了 Rocketship 精灵，则停止脚本运行。

8.3.15 15. 游戏 - 吃苹果

在接下来的项目中，我们将在 PictoBlox 中玩一些有趣的小游戏。

在这里，我们使用按键控制甲壳虫去吃苹果的游戏。

当绿旗被点击时，按下按键，甲壳虫将旋转，再次按下按键，甲壳虫停止运行，并按照这个角度向前走。你需要控制甲壳虫的角度，让它在前进的同时不碰到地图上的黑线，直到吃到苹果。如果碰到黑线，游戏结束。



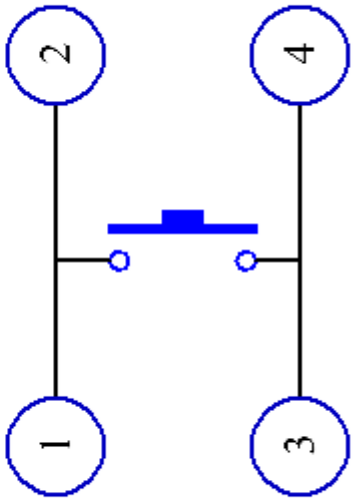
你将学习

- 触碰颜色事件
- 嵌套条件循环
- 切换背景和绘制背景

搭建电路

该按钮为 4 引脚器件，因为 1 引脚连接到 2 引脚，3 引脚连接到 4 引脚，当按下按钮时，4 个引脚连接在一起，从而闭合电路。

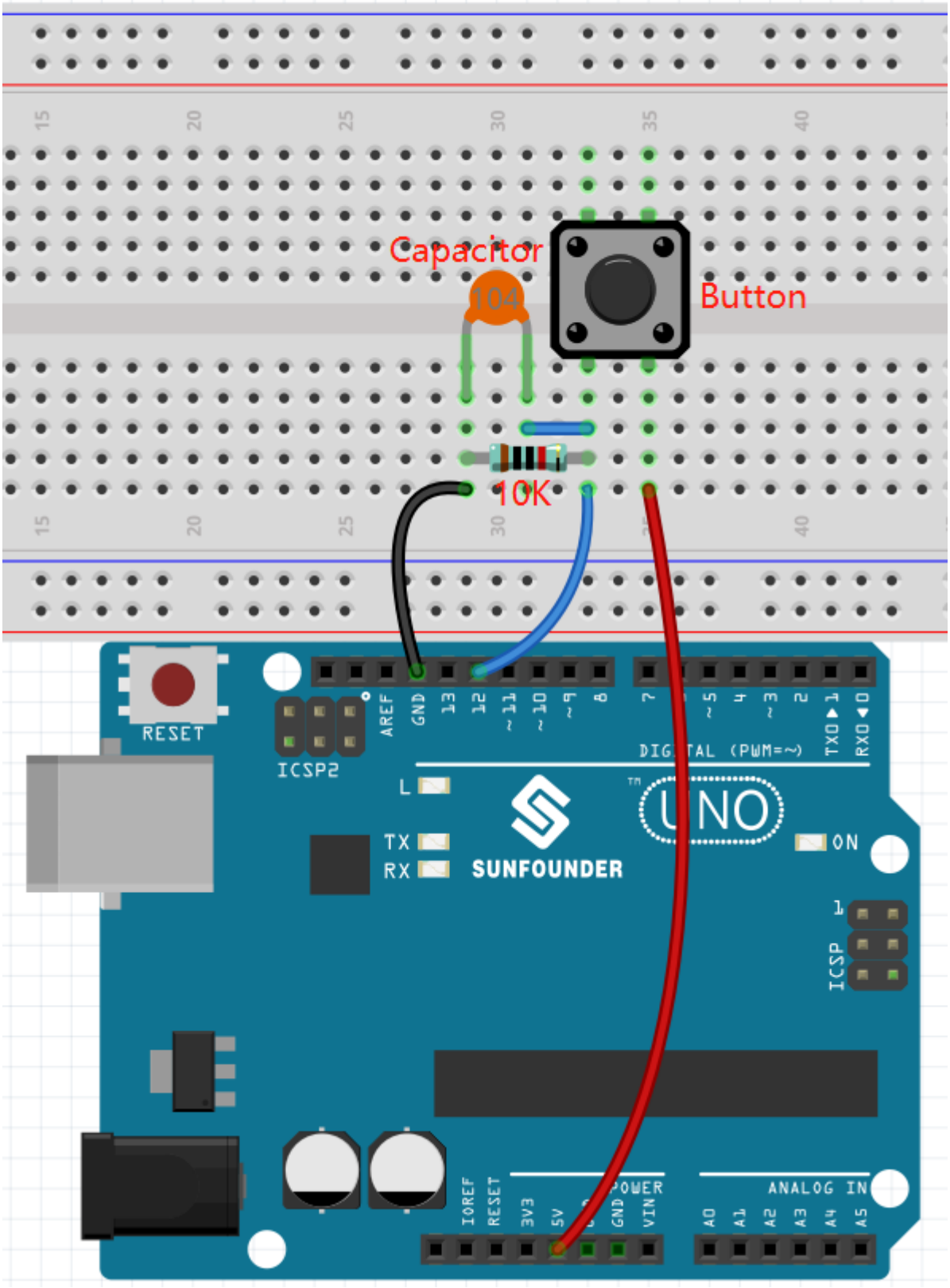
Button



Internal Structure

按照下图搭建电路：

- 将按钮左侧的其中一个引脚连接到 12 引脚，该引脚连接下拉电阻和 0.1uF（104）电容（以消除抖动并在按钮工作时输出稳定电平）。
- 将电阻和电容的另一端连接到 GND，将按钮右侧的一个引脚连接到 5V。


Capacitor
104
Button
10K
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
VIN
A0
A1
A2
A3
A4
A5

- 面包板
- 按键

- 电阻
- 电容

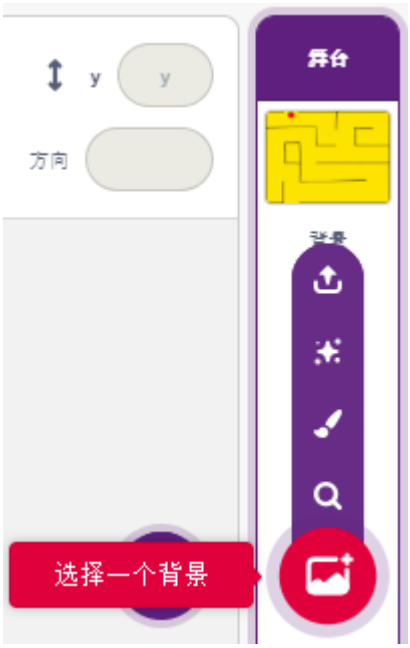
编程

我们要实现的效果是用按键的控制 Beetle 精灵前进方向，在不碰触 Maze 背景上的黑色线的情况下，吃到苹果，吃到后会切换背景。

现在来添加相关背景和精灵。

1. 添加背景

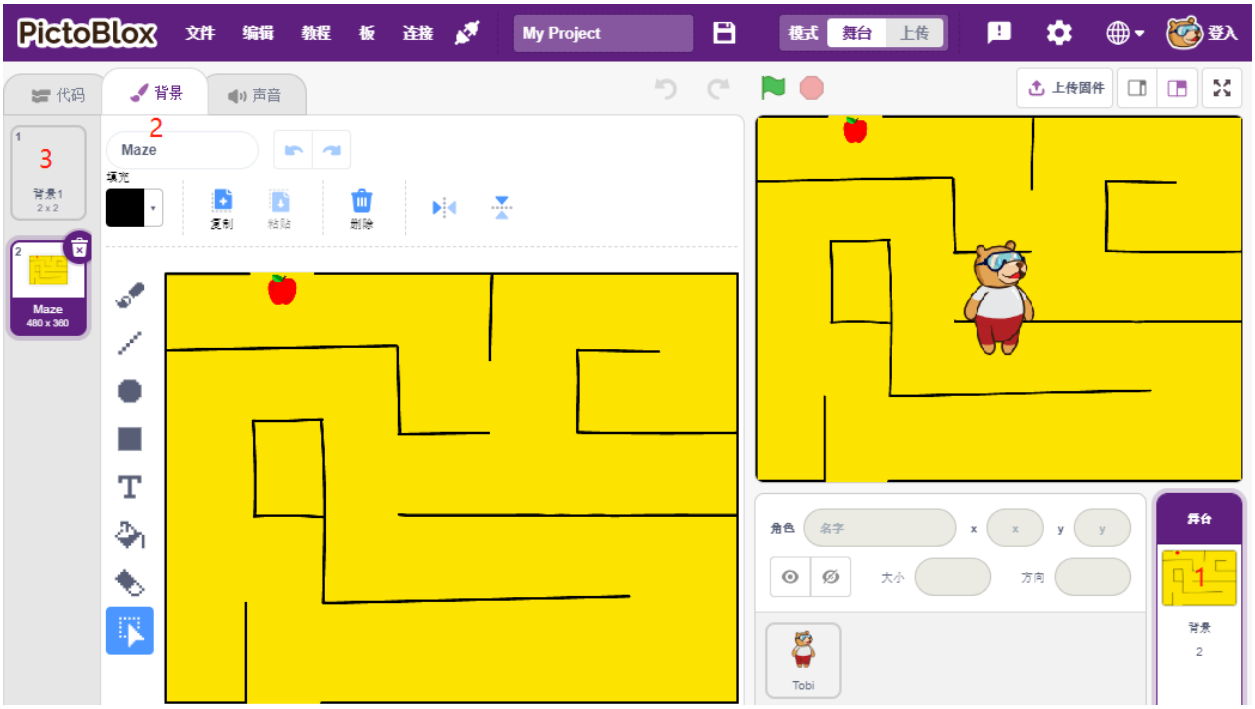
通过 **选择一个角色**按钮添加一个 **Maze** 背景。



2. 画一个背景

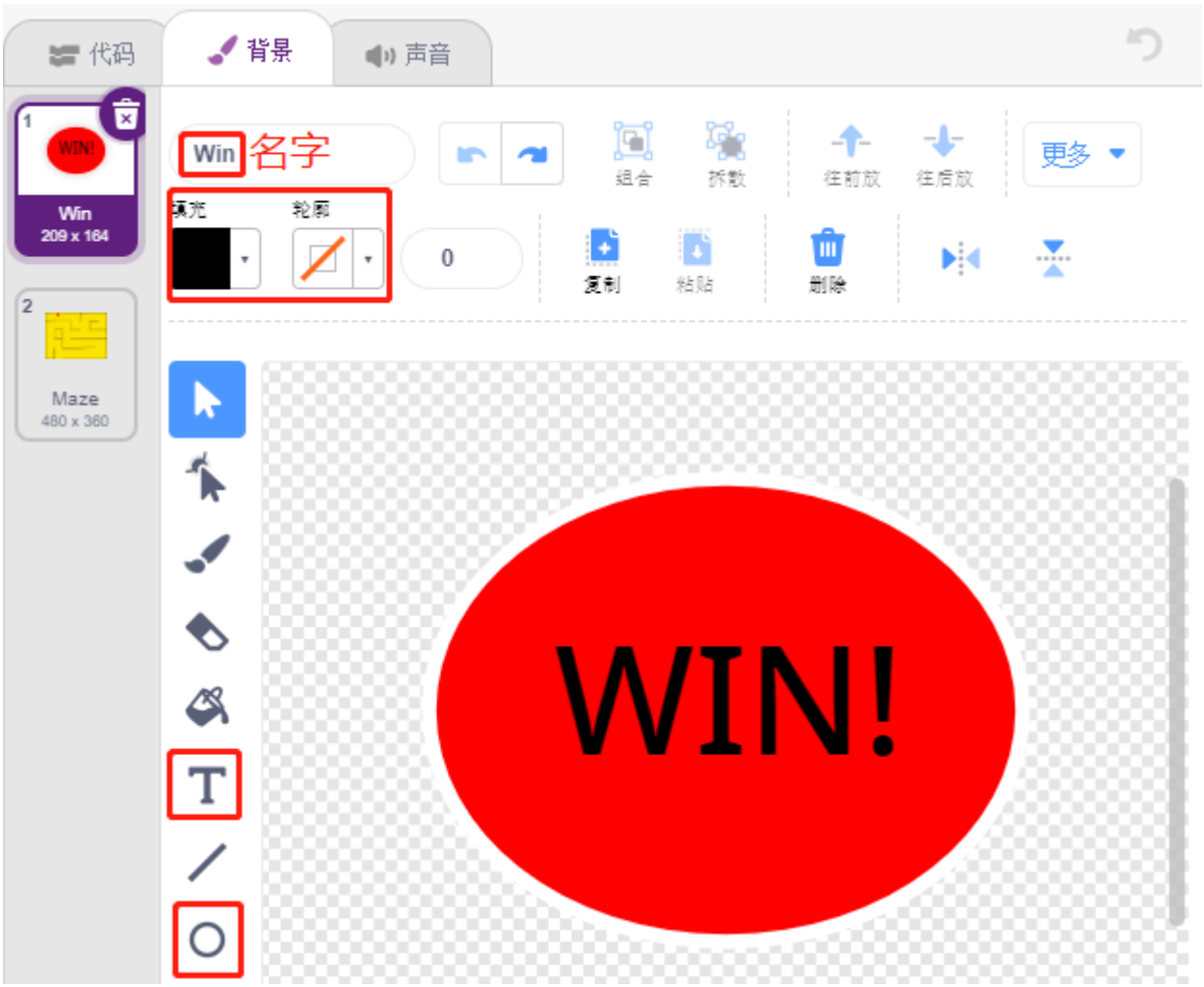
我们需要一个背景来提示 beetle 精灵已成功吃到 Maze 背景上的苹果。现在来简单绘制一个背景，在背景上出现 WIN! 字符。

首先点击背景缩略图，进入到 **背景**页面，点击空白的 **背景 1**。



现在开始绘制，你可以参考下图绘制，也可以自行绘制一个背景，只要表达的意思是赢了就行。

- 使用画圆工具，画一个椭圆，颜色设置为红色，没有轮廓。
- 再使用文本工具，写字符 WIN!, 将字符颜色设置为黑色，调整字符的大小和位置。
- 将背景命名为 Win



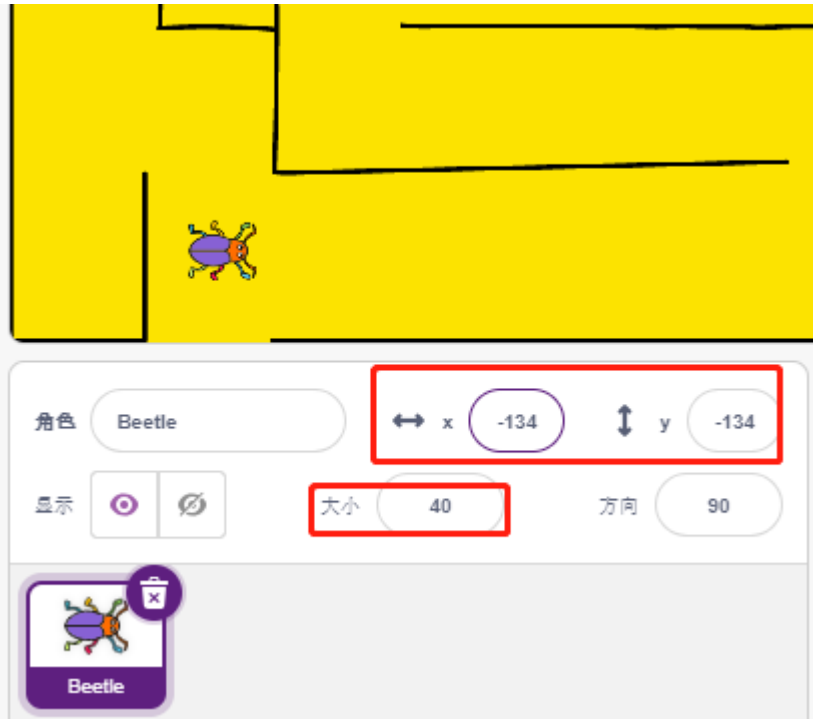
3. 为背景编写脚本

在每次游戏启动的时候，都需要将背景切换为 Maze。

4. 添加一个精灵

删除默认精灵，选择 **Beetle** 精灵.

将 Beetle 精灵放在 Maze 精灵的入口处，记住此时的 x,y 坐标值，并调整精灵大小为 40%。



5. 为精灵 Beetle 编写脚本

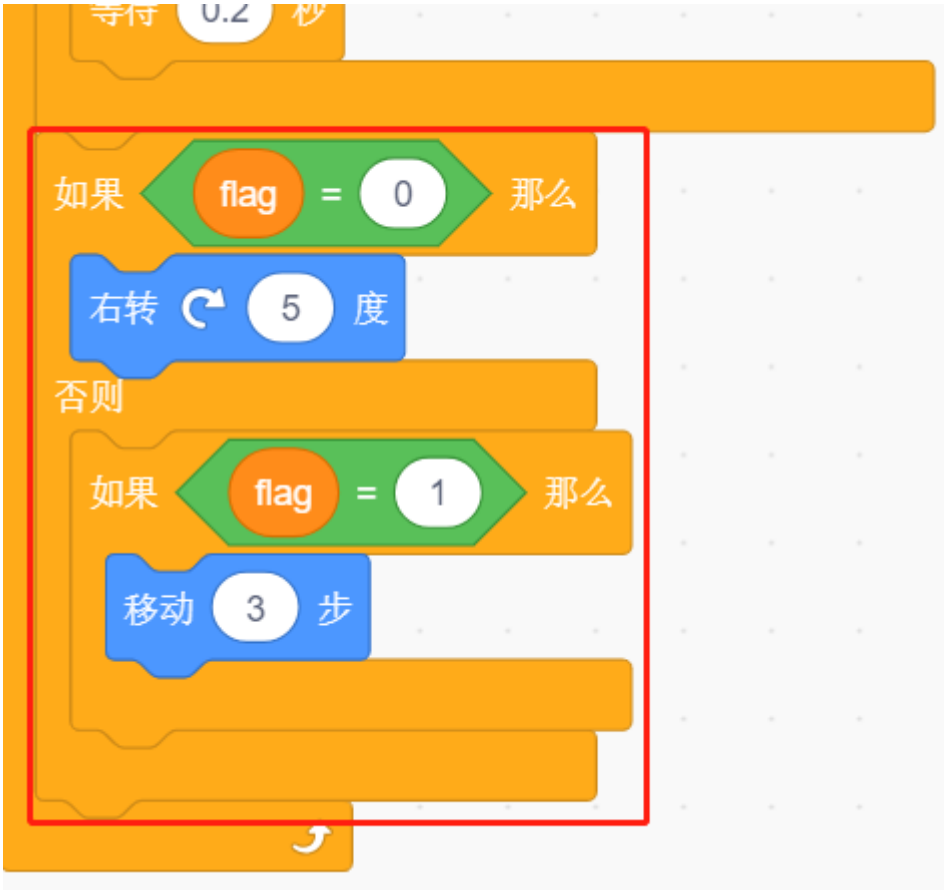
现在为精灵 Beetle 编写脚本，让它能够在按键的控制下前进和转动方向，工作流程如下：

- 当绿色旗子点击时，将 Beetle 角度设置为 90, 位置为（-134，-134），或者换成你自己放置的位置的坐标值。创建变量 flag，并设置初始值为-1。

接下来在 [重复执行] 块中，用了 4 个 [如果 () 那么] 块来判断各种可能出现的情况。

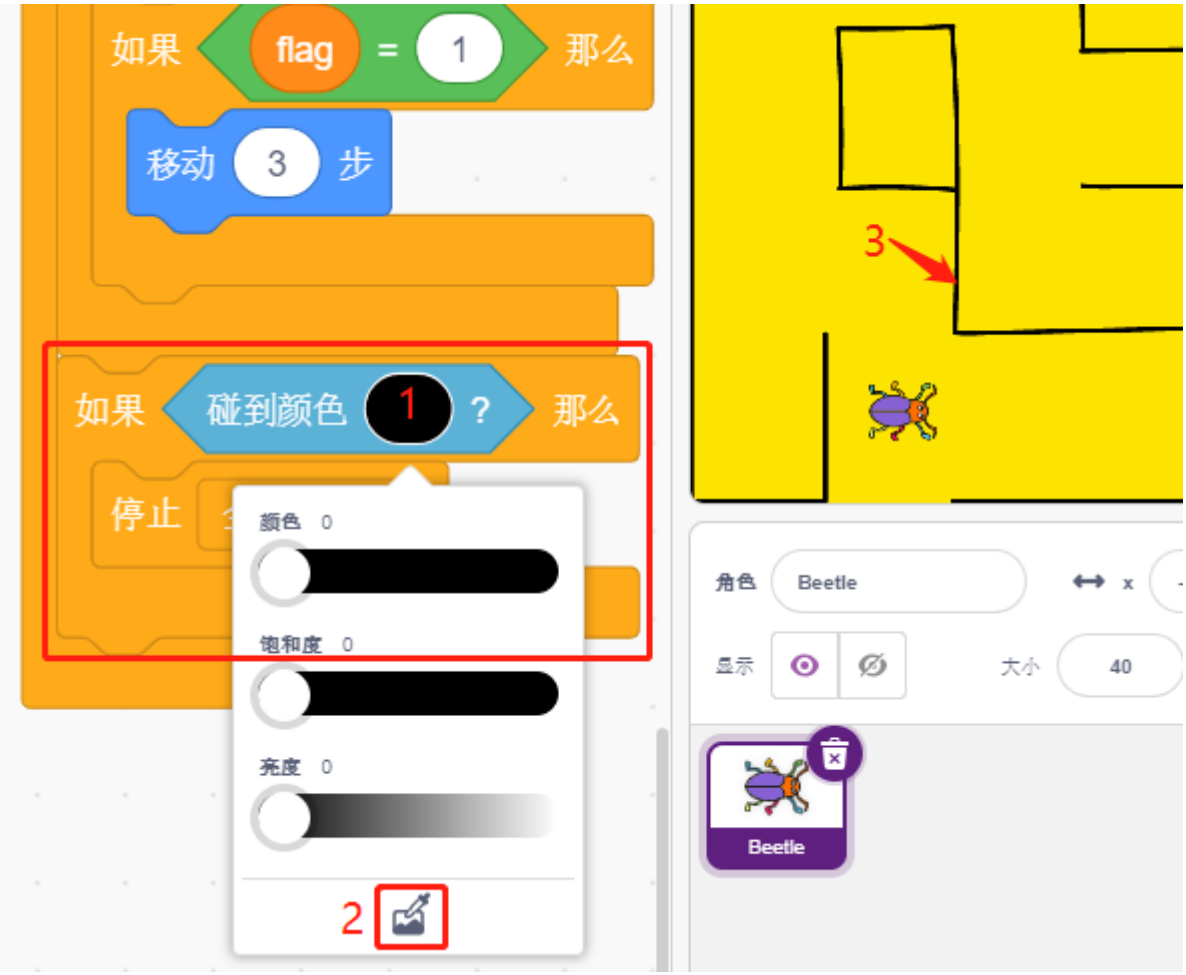
- 如果按键为 1（被按下），用 [除以 () 的余数] 块将变量 flag 的值在 0 和 1 之间切换（这次按下为 0，下次按下为 1，以此交替）。

- 如果 flag=0（此次按键按下），让 Beetle 精灵顺时针转动。再判断 flag 是否等于 1（按键再次按下），Beetle 精灵前进。否则就一直顺时针转动。

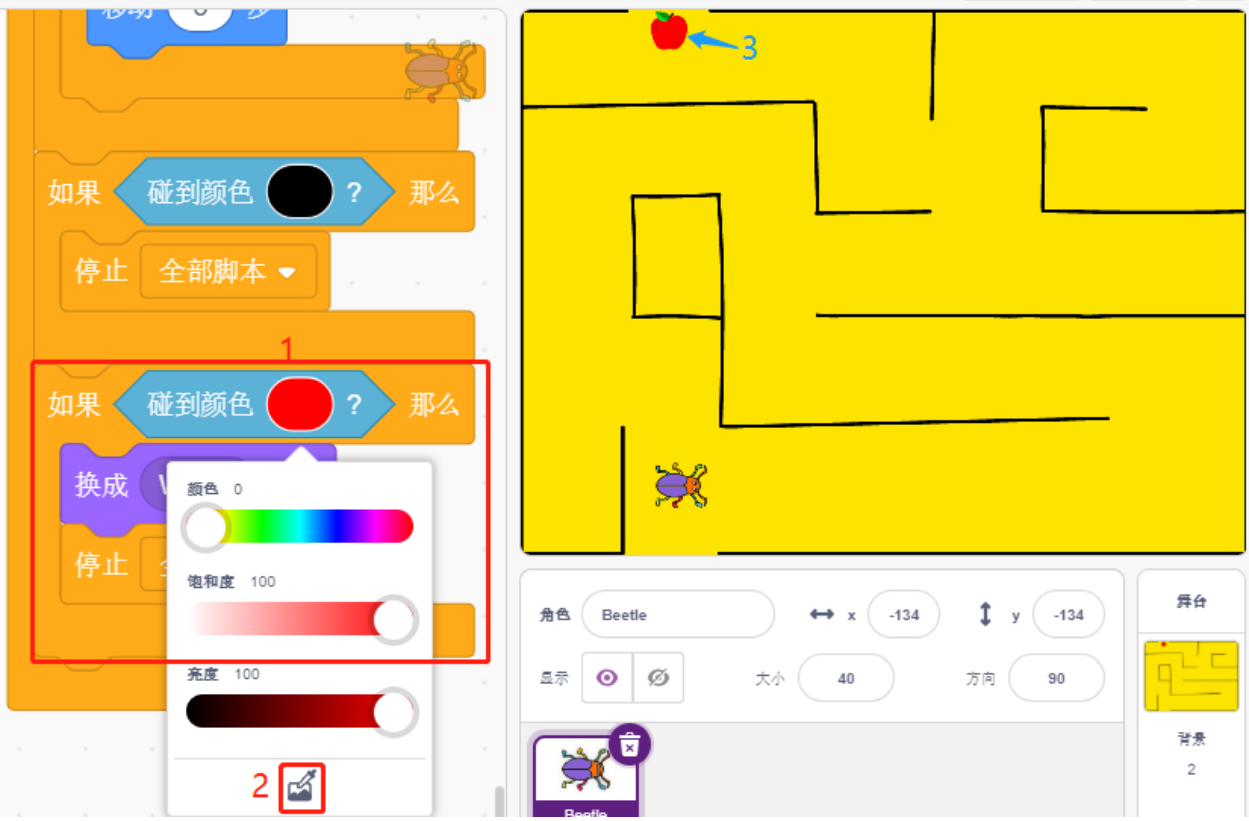


- 如果 **Beetle** 精灵碰到黑色（在 Maze 背景上的黑色线），游戏结束并且脚本停止运行。

备注： 你需要点击 [碰到颜色 () ?] 块中的颜色区域，然后用吸管工具来吸取舞台上黑线的颜色。如果你随意选择黑色，则此 [碰到颜色 () ?] 块将不起作用。



- 如果甲壳虫碰到红色（也用吸管工具吸取苹果的红色），背景将切换到 Win，这意味着游戏成功并停止脚本运行。



8.3.16 16. 游戏 - 愤怒的鹦鹉

在这里，我们使用超声波模块来玩一个愤怒的鹦鹉的游戏：

脚本运行后，绿色的竹子会在最右侧随机高度慢慢向左移动。现在将手放在超声波上方，如果手与超声波距离小于 10，鹦鹉将向上飞，否则将向下落。你需要控制你的手与超声波的距离，让鹦鹉可以躲过绿色的竹子，如果碰到了，则游戏结束。

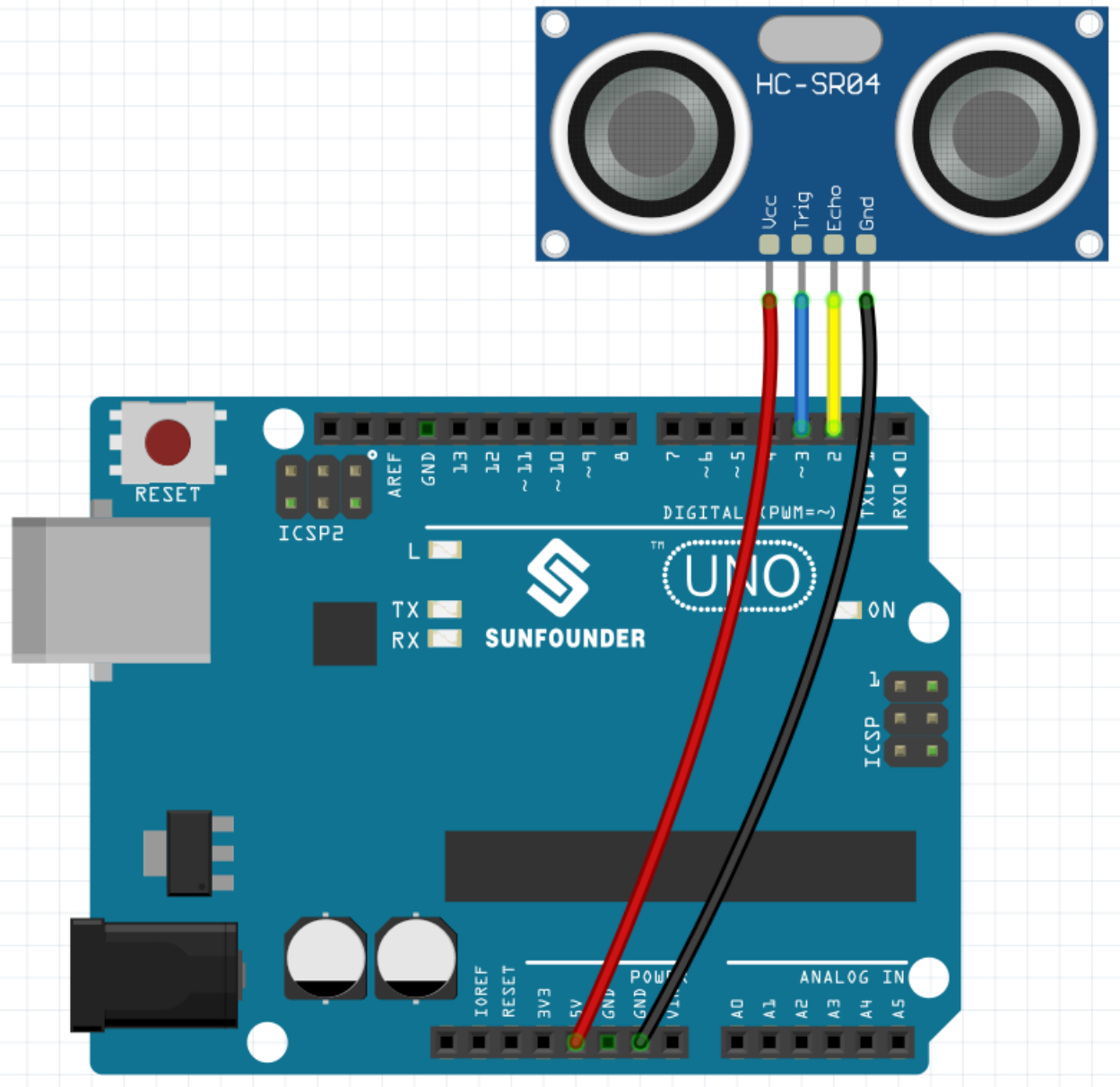
你将学习

- 超声波模块工作原理
- 克隆精灵本身

搭建电路

超声波传感器模块是一种使用超声波测量与物体之间距离的仪器。它有两个探头。一种是发送超声波，另一种是接收超声波并将发送和接收的时间转换为距离，从而检测设备与障碍物之间的距离。

现在根据下图构建电路：



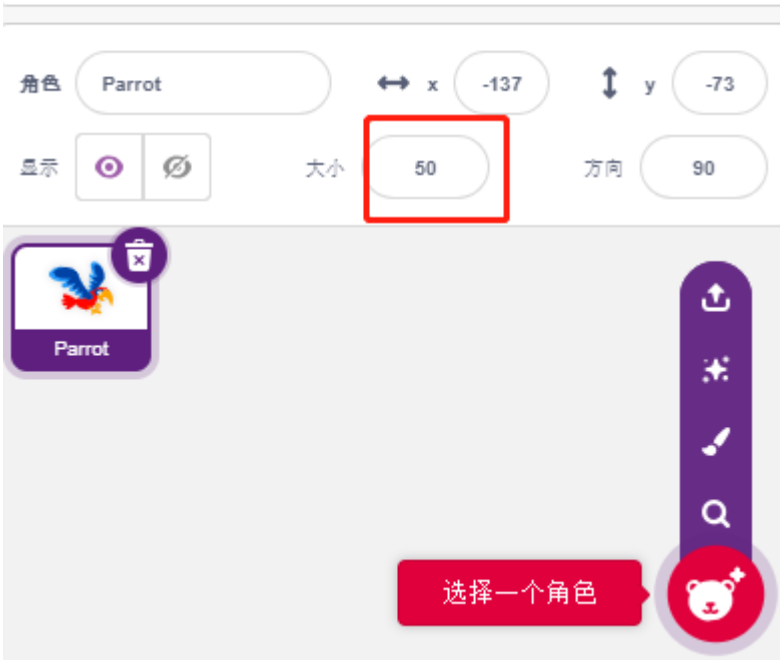
- 面包板
- 超声波模块

编程

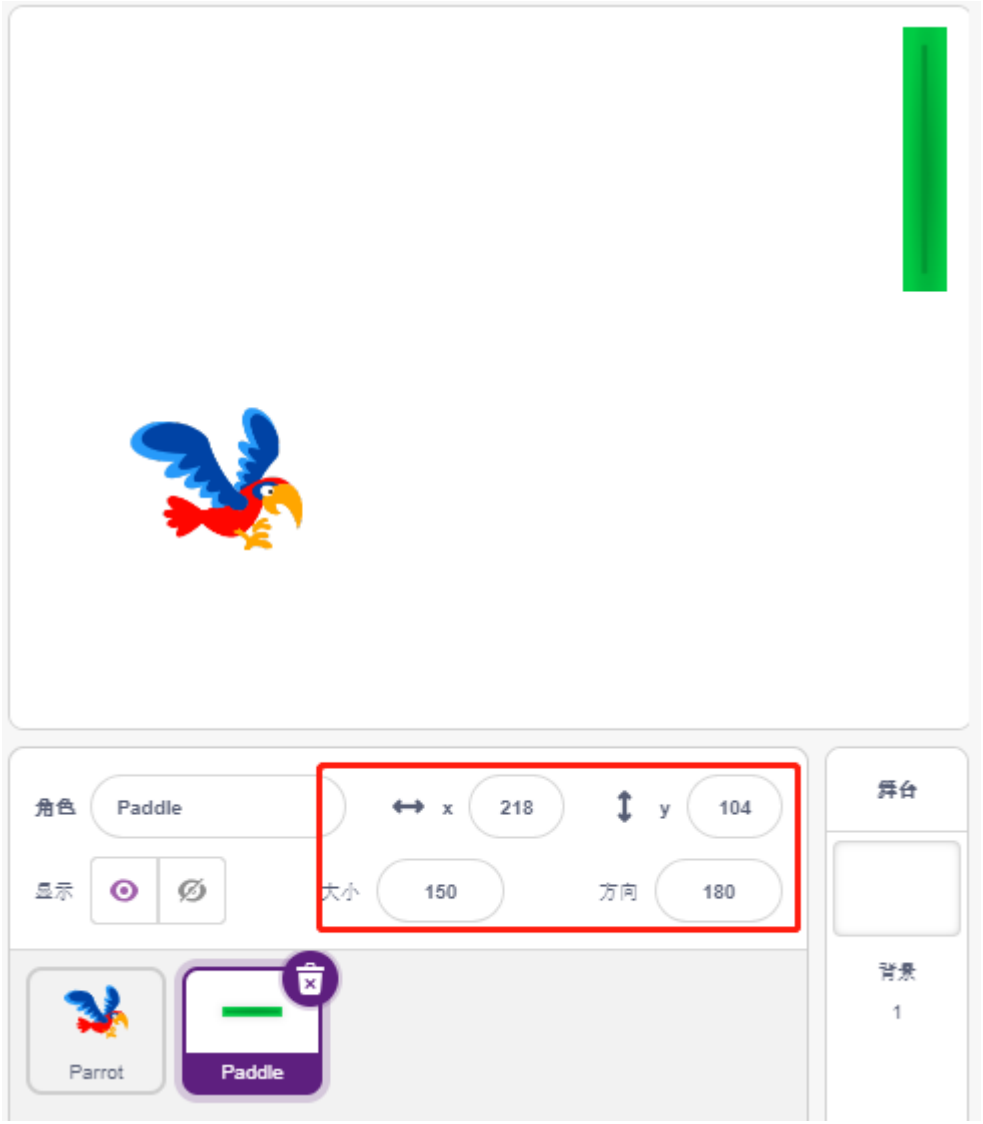
我们要实现的效果是用超声波控制精灵 Parrot 的飞行高度，同时躲开 Paddle 精灵。

1. 添加精灵

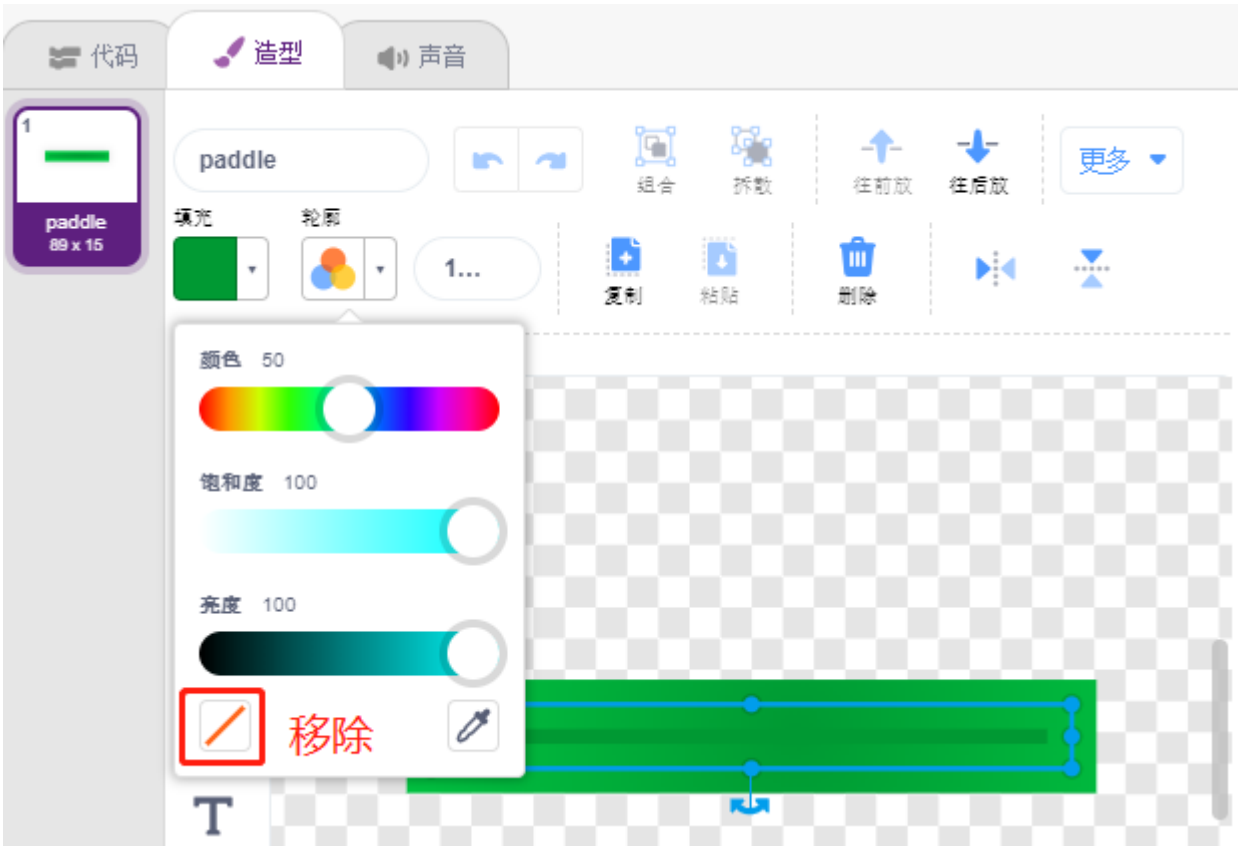
删除默认精灵，并使用 **选择一个角色**按钮添加鹦鹉精灵。将其大小设置为 50%，并将其位置移动到左中心。



现在添加 Paddle 精灵，设置它的尺寸为 150%，将它的角度设置为 180，并将它的初始位置挪到右上角。



进入到 Paddle 精灵的 **造型**页面，移除轮廓.



2. 为 Parrot 编写脚本

现在为 Parrot 精灵脚本，它处于飞行状态，飞行高度由超声波模块的决定，工作流程如下：

- 当绿色旗子被点击时，每隔 0.2s 切换精灵造型，让它一直处于飞行状态。

- 读取超声波传感器的值，用 [四舍五入] 块取整之后，存放到变量 distance 中。

- 如果超声波检测的距离低于 10cm, 则让 y 坐标增加 50，即 Parrot 精灵将向上飞. 否则，y 坐标值减少 20，即 Parrot 向下落。

- 如果 Parrot 精灵碰到了 Paddle 精灵，则游戏结束，脚本停止运行。

3. 为 Paddle 精灵编写脚本

现在为 Paddle 精灵编写脚本，它需要随机出现在舞台上。工作流程如下：

- 当绿色旗子被点击时，隐藏精灵 Paddle, 并同时将它自己克隆。[克隆（）] 块是一个控制块和一个堆栈块。它在参数中创建了精灵的克隆。它还可以克隆它正在运行的精灵，递归地创建克隆的克隆。

- 当 Paddle 是以克隆体出现，显示它的位置，它的 x 坐标为 220（最右边），y 坐标在（-125~125）随机（高度随机）。

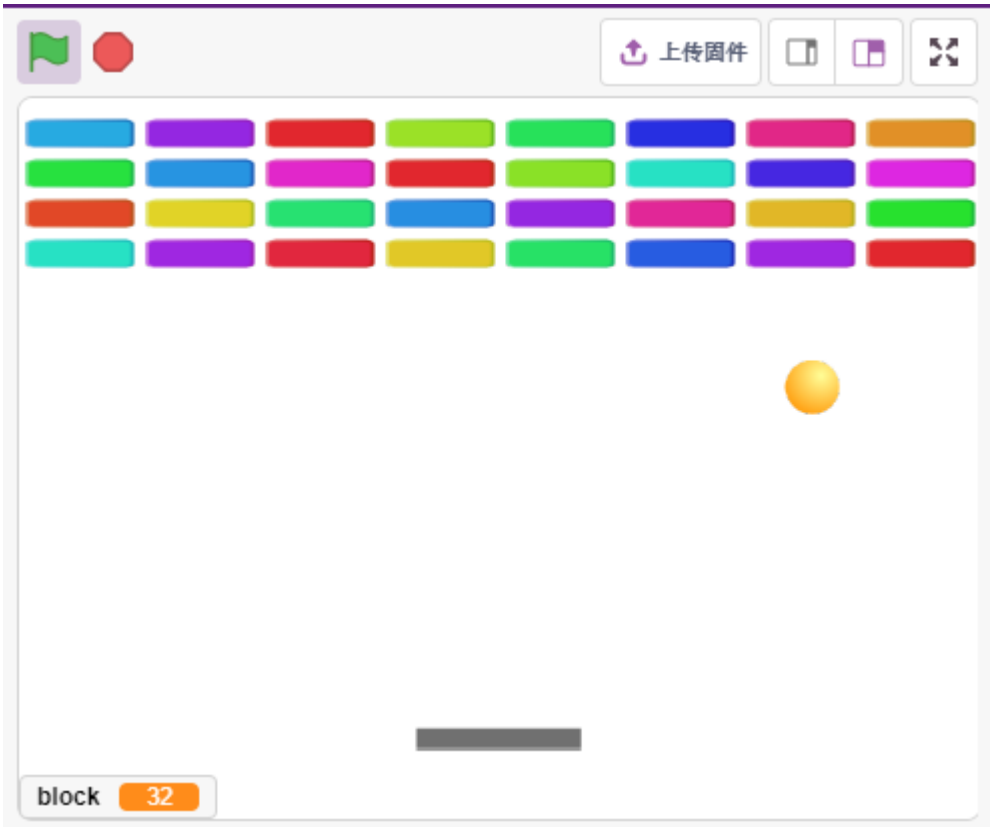
- 使用 [重复执行 () 次] 块让它的 x 坐标值慢慢减少，这样你就能看到 Paddle 精灵的克隆体从最右边慢慢移动到最左边，直到消失。

- 重新克隆一个新的 Paddle 精灵，并将上一个克隆的 Paddle 删除。

8.3.17 17. 游戏 - 打砖块

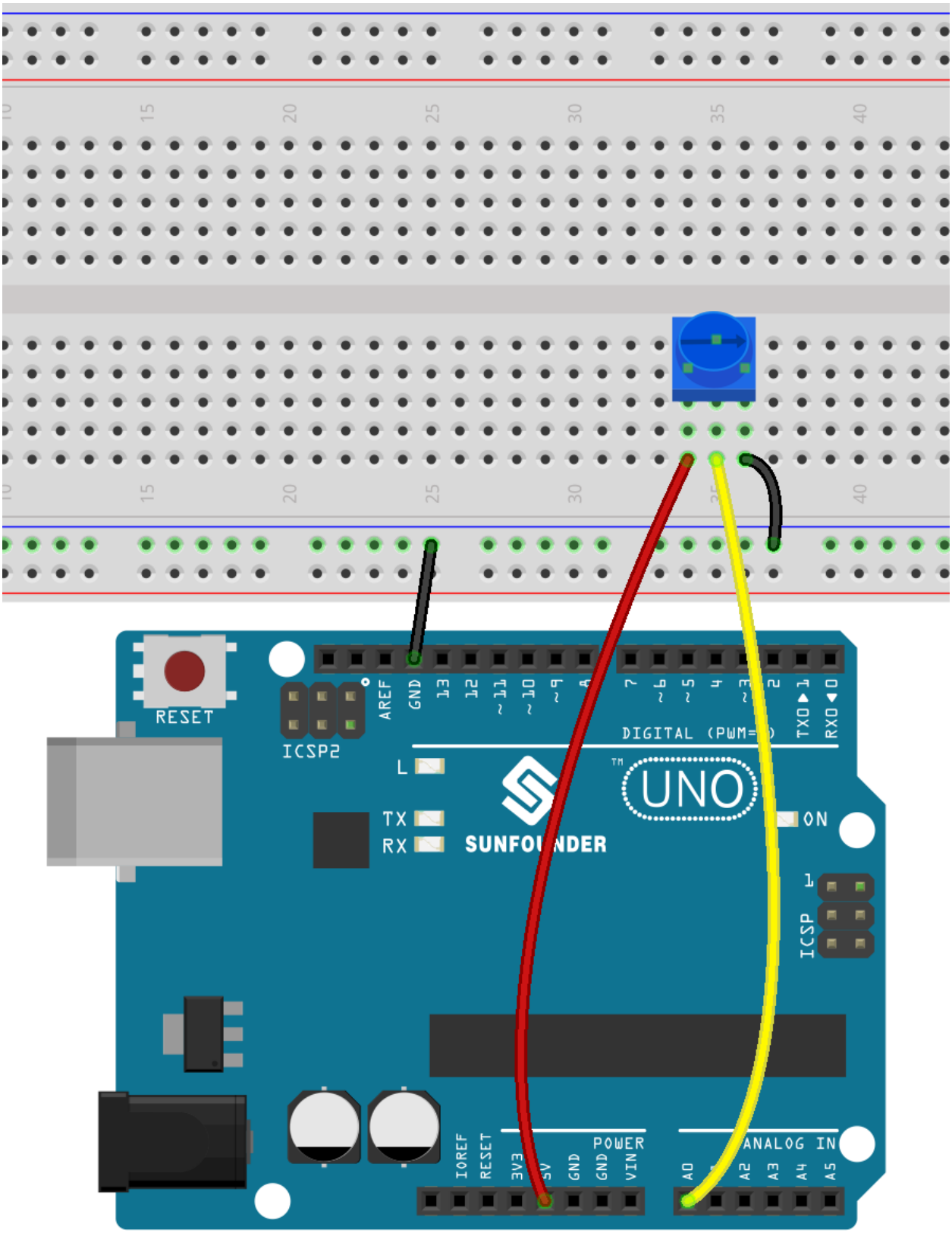
在这里，我们使用电位器来玩一个打砖块的游戏：

脚本运行后，你需要移动舞台下方的横条，让黄色的球在落下的时候能够接住，并让它向上弹向舞台上方的砖块，碰到一个砖块，砖块就会消失。砖块全部消失，则游戏赢了，若横条没有接住球，则游戏结束。



搭建电路

按照下图搭建电路，电位器是一个有 3 个端子的电阻元件，2 侧的引脚分别接 5V 和 GND，中间引脚接到 A0，经过 Arduino 板的 ADC 转换器转换后，得到的数值范围为 0-1023.



- 面包板

- 电位器

编程

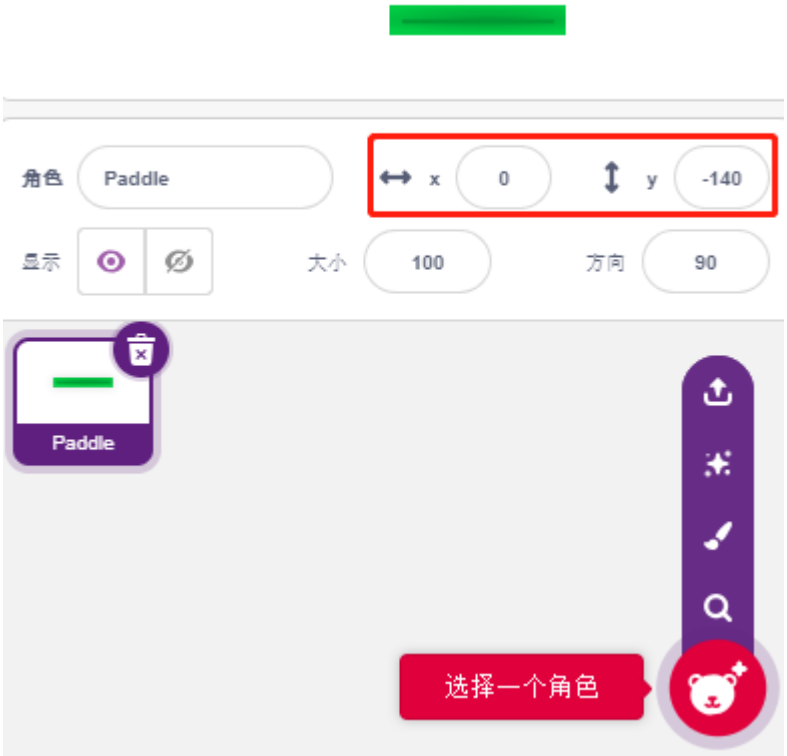
在舞台上有 3 个精灵：

- Paddle 精灵：Paddle 精灵初始位置在舞台的下方中心，它的左右移动由电位器控制。
- Ball 精灵：Ball 精灵在舞台上四处弹，当向上弹碰到 Block1 精灵，它将向下反弹；下落的时候如果碰到了 Paddle 精灵，它将向上弹，若没有碰到，则停止脚本运行。
- Block1 精灵：在舞台上方克隆 4x8 个 Block1 精灵，颜色随机，当 Block1 的克隆被 Ball 精灵碰到，则删除该克隆。

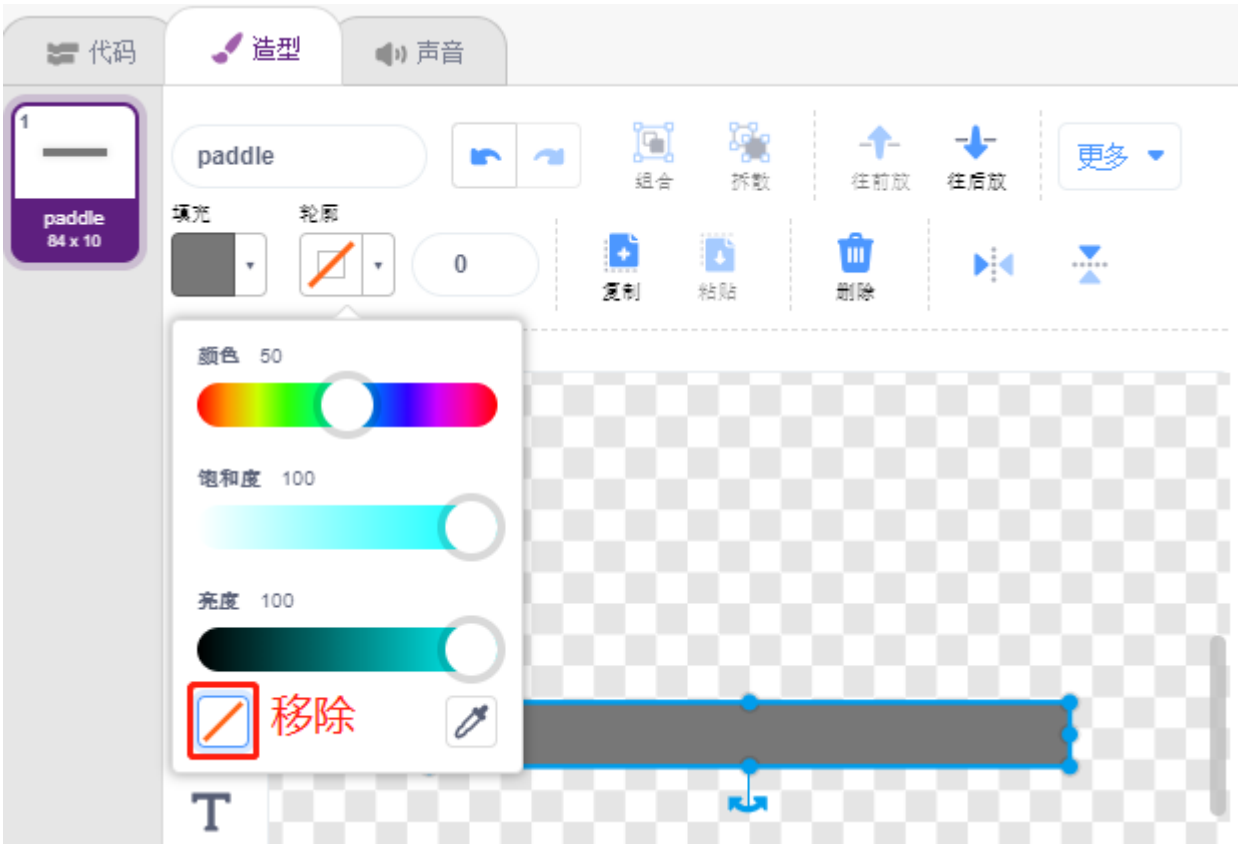
1. Paddle 精灵

Paddle 要实现的效果是，初始位置为舞台下方中间，由电位器来控制它向左或向右移动。

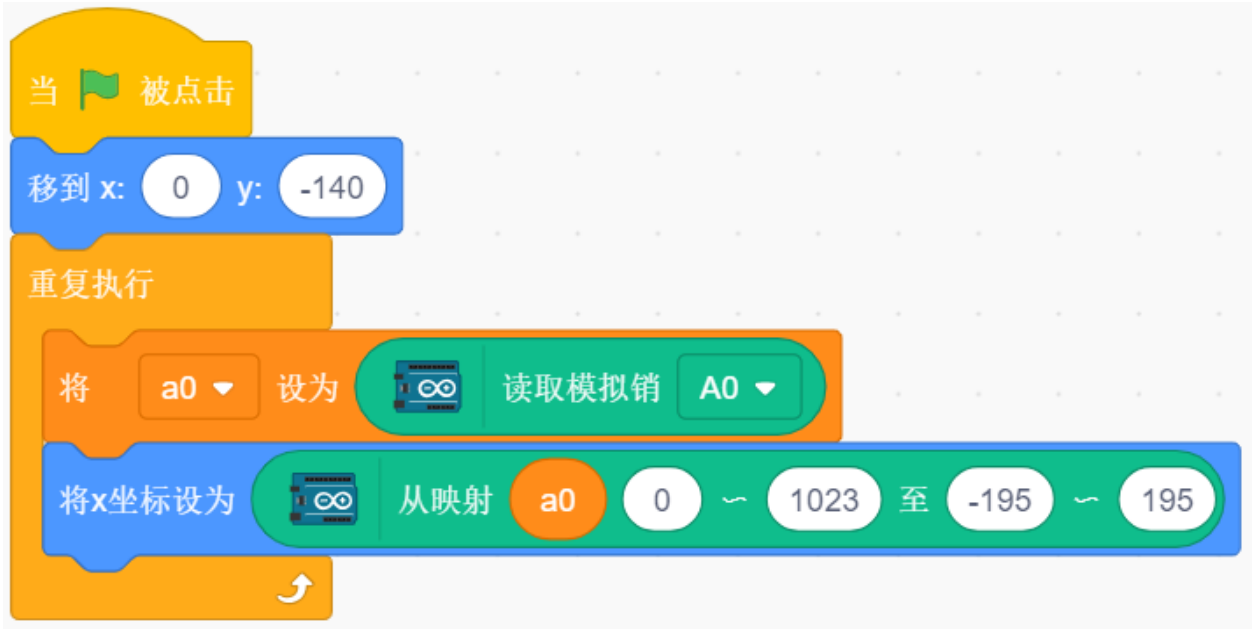
- 删除默认精灵，使用 **选择一个角色**按钮来添加 **Paddle** 精灵，并将它的 x 和 y 设置为（0，-140）。



- 进入到 **造型**页面，移除轮廓并将它的颜色修改成深灰色。



- 现在为 Paddle 精灵编写脚本，当绿旗点击时，设置它的初始位置为（0，-140），读取 A0（电位器）的值存放到变量 a0. 由于 Paddle 精灵在舞台上从左边移动到右边的 x 坐标为-195~195，所以需要用 [从映射 ()] 块将变量 a0 范围 0~1023 映射到-195~195。

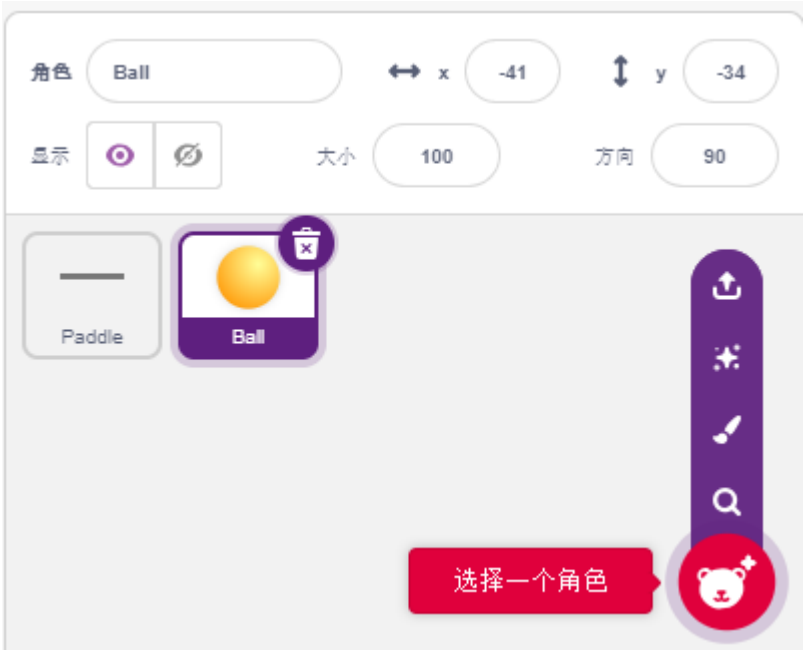


- 现在你可以旋转电位器，看起 Paddle 是否能在舞台上左右移动。

2. Ball 精灵

Ball 精灵要实现的效果是：在舞台上向移动，碰到边缘就反弹；如果碰到舞台上方的 block 时就会向下弹；如果下落过程中碰到 Paddle 精灵就会向上弹；如果在下落过程中，没有碰到 Paddle 精灵，停止脚本运行，游戏结束。

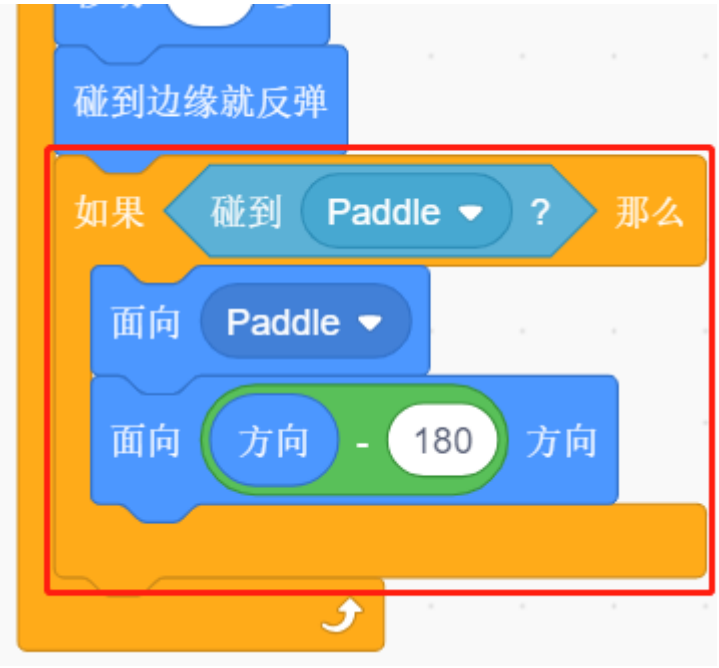
- 添加 **Ball** 精灵。

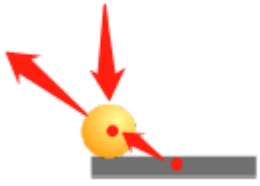


- 当绿旗点击时，设置 Ball 精灵的角度为 45°，并设置初始位置为（0，-120）.

- 现在让 Ball 精灵在舞台上移动，碰到边缘则反弹，你可以点击绿旗看下效果。

- 当 Ball 精灵碰到 Paddle 精灵时，进行一次反射。简单的做法是让角度直接取反，但是这样做的话，你会发现小球的路径是完全固定的，这未免过于无趣。因此，我们借助两个精灵的中心点进行计算，让小球往挡板中心的反方向弹出。





- 当 Ball 精灵下落到舞台边缘，则停止脚本运行，游戏结束。

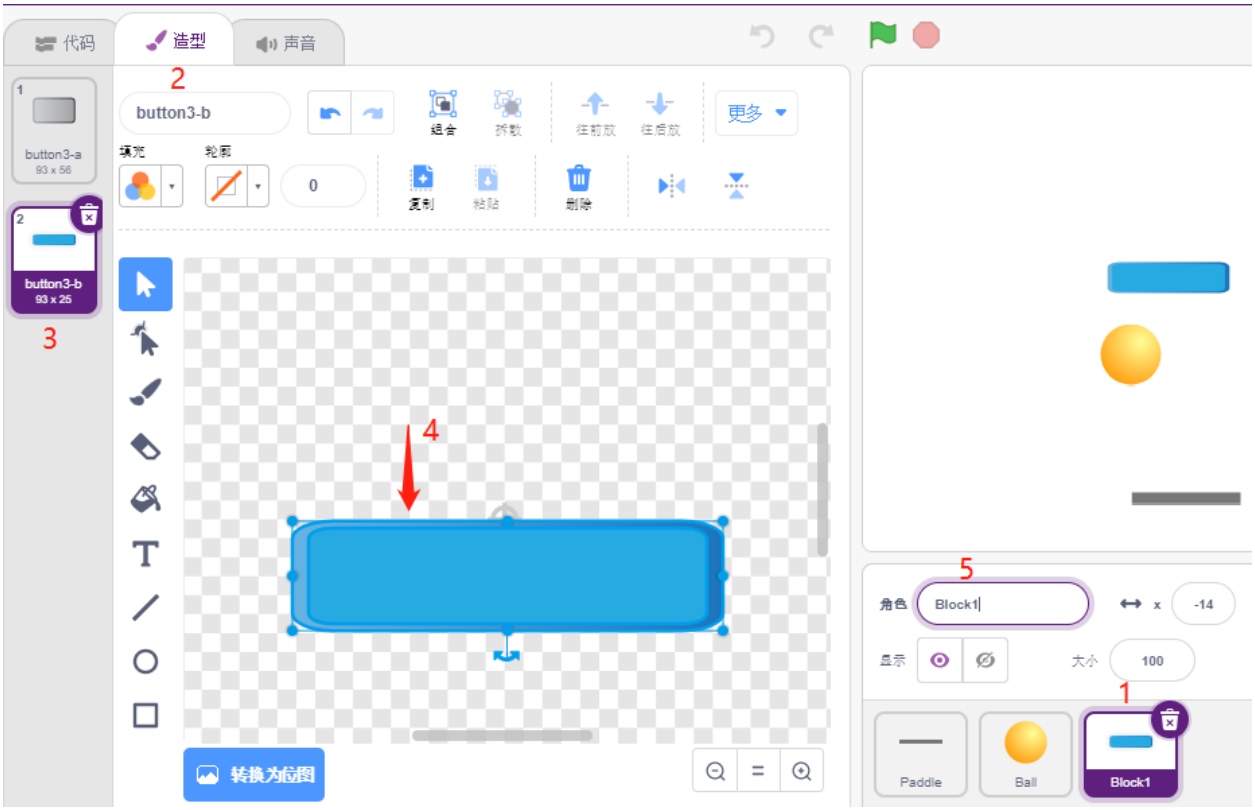
3. Block1 精灵

Block1 精灵要出现的效果是，在舞台上方克隆 4x8 个它自己，颜色随机显示，如果某个克隆体被 Ball 精灵碰到，则删除这个克隆。Block1 精灵在 PictoBlox 库中没有，你需要自己绘制或者用现有的精灵修改。这里我们是将用 **Button3** 精灵修改。

- 添加 **Button3** 精灵，进入到 **造型**页面，删除 button-a，将 button-b 宽度和高度都缩小，并将精灵名字改为 **Block1** , 如下图所示：

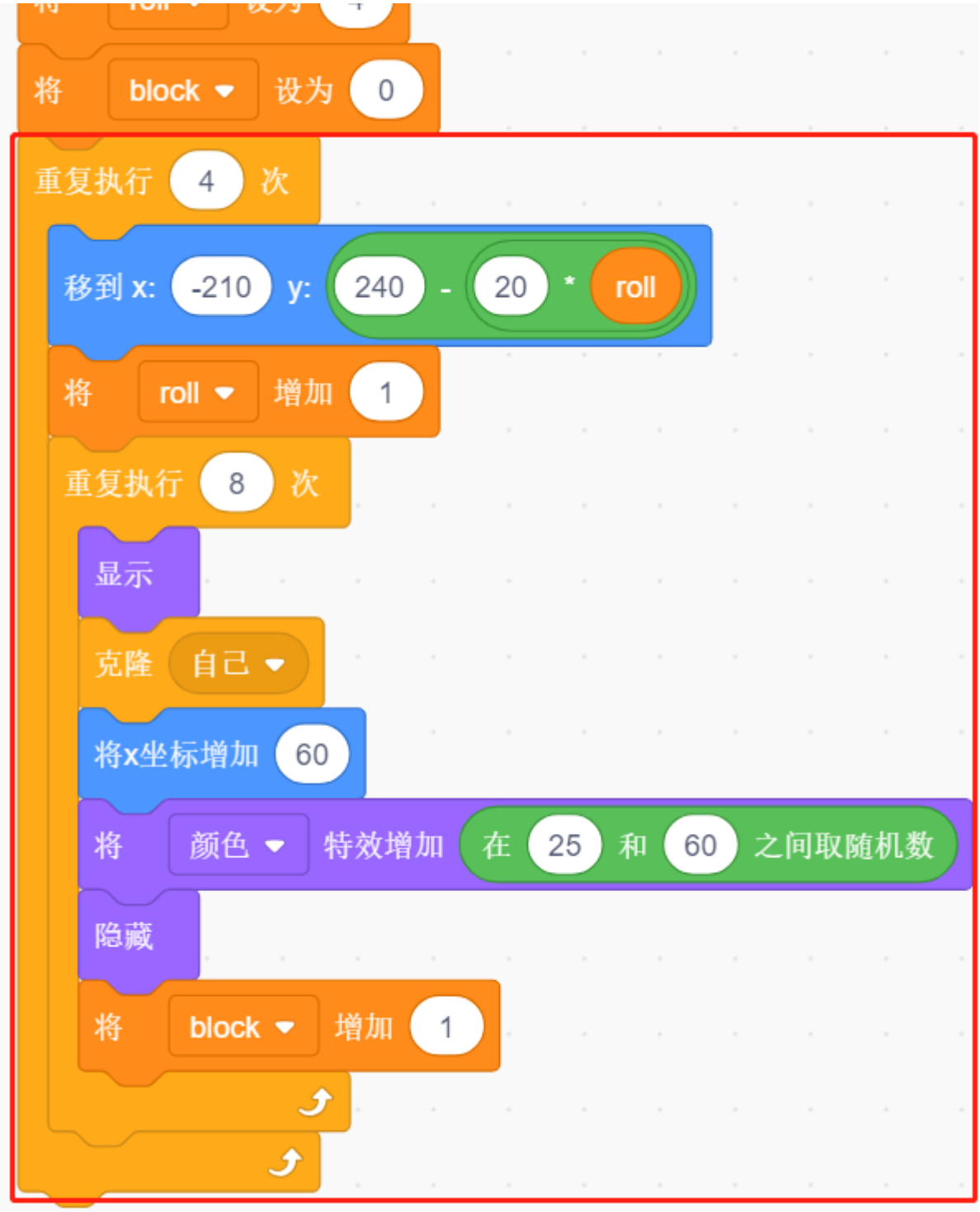
备注:

- 对于 Block1 的宽度，你可以大概在屏幕上模拟下，看下是否一行能放下 8 个，如果不能再适当缩小宽度。
- 在缩小 Block1 精灵的过程中，需要让中心点保持在精灵的中间位置。

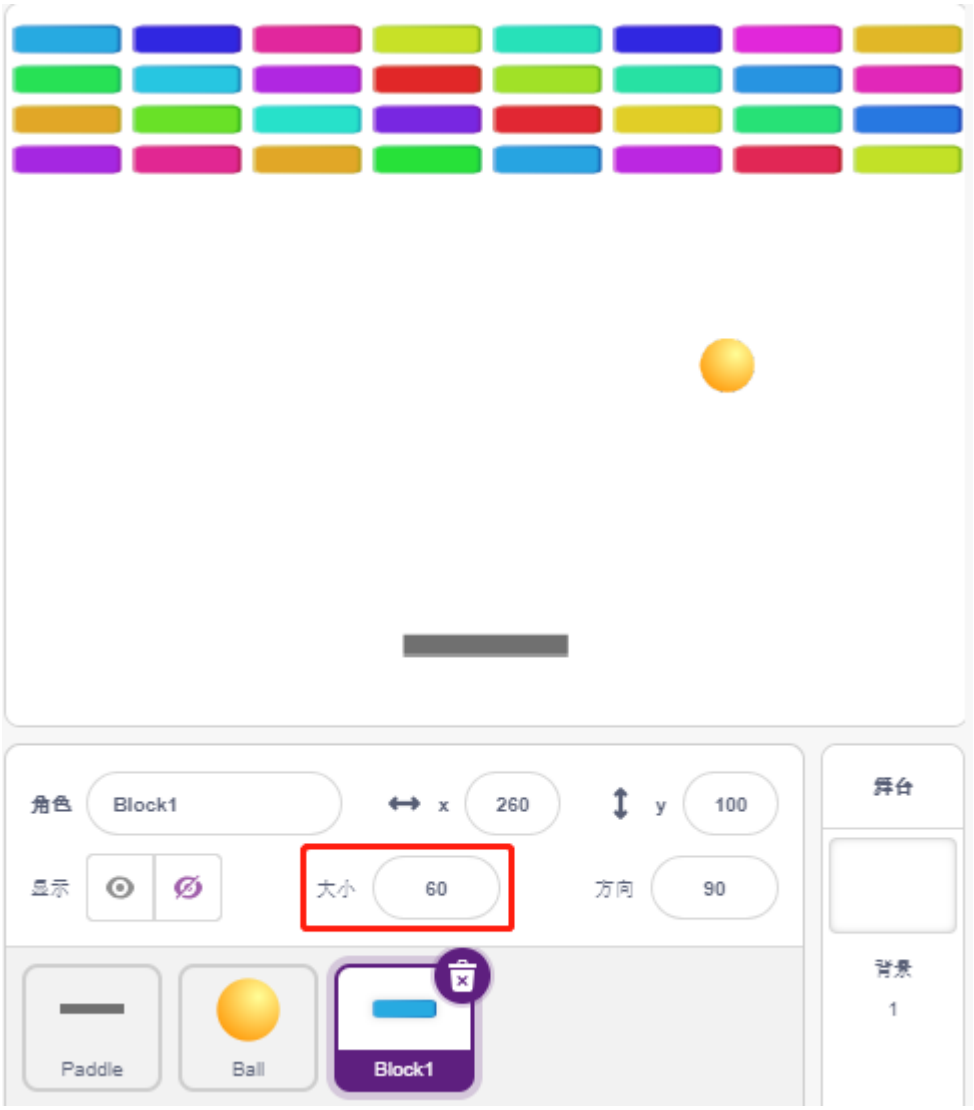


- 现在开始编写脚本，先创建 2 个变量，block 用来存放块的个数，roll 用来存放行数。

- 我们需要让克隆 Block1 精灵，让它从左到右，从上到下，一个一个显示，总共显示 4x8 个，颜色随机显示。



- 脚本写完后，点击绿旗，看下舞台上的显示效果，如果太紧凑或者太小，你可以改变 **大小**的值.



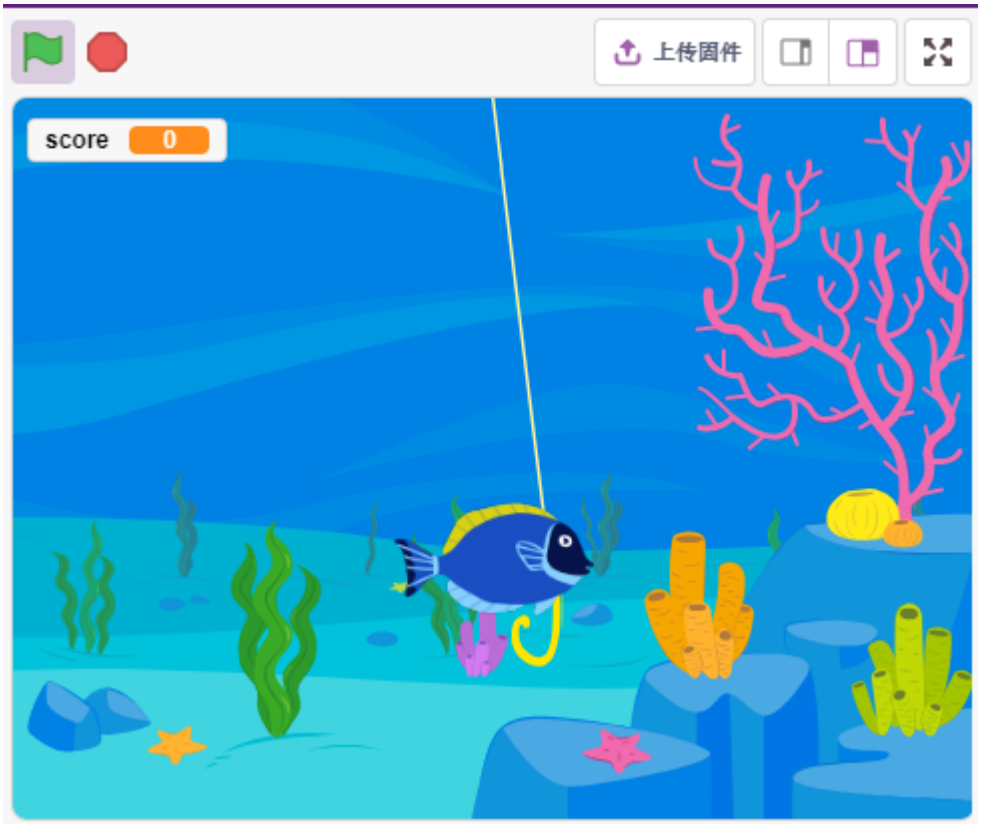
- 现在编写触发事件。如果克隆的 **Block1** 精灵接触到 **Ball** 精灵，则删除该克隆并广播消息 **crush**。

- 回到 **Ball** 精灵，当接收到广播 **crush** (即 **Ball** 精灵碰到 **Block1** 的克隆)，则 **Ball** 从反方向弹出。

8.3.18 18. 游戏 - 钓鱼

在这里，我们使用按键来玩一个钓鱼的游戏：

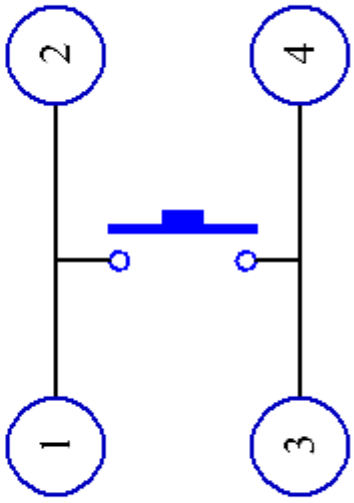
脚本运行后，鱼在舞台上左右游动，你需要在鱼快靠近鱼钩时，按下按键 (建议按久一点时间) 来将鱼钓起，同时会自动记录钓鱼的数量。



搭建电路

该按钮为 4 引脚器件，因为 1 引脚连接到 2 引脚，3 引脚连接到 4 引脚，当按下按钮时，4 个引脚连接在一起，从而闭合电路。

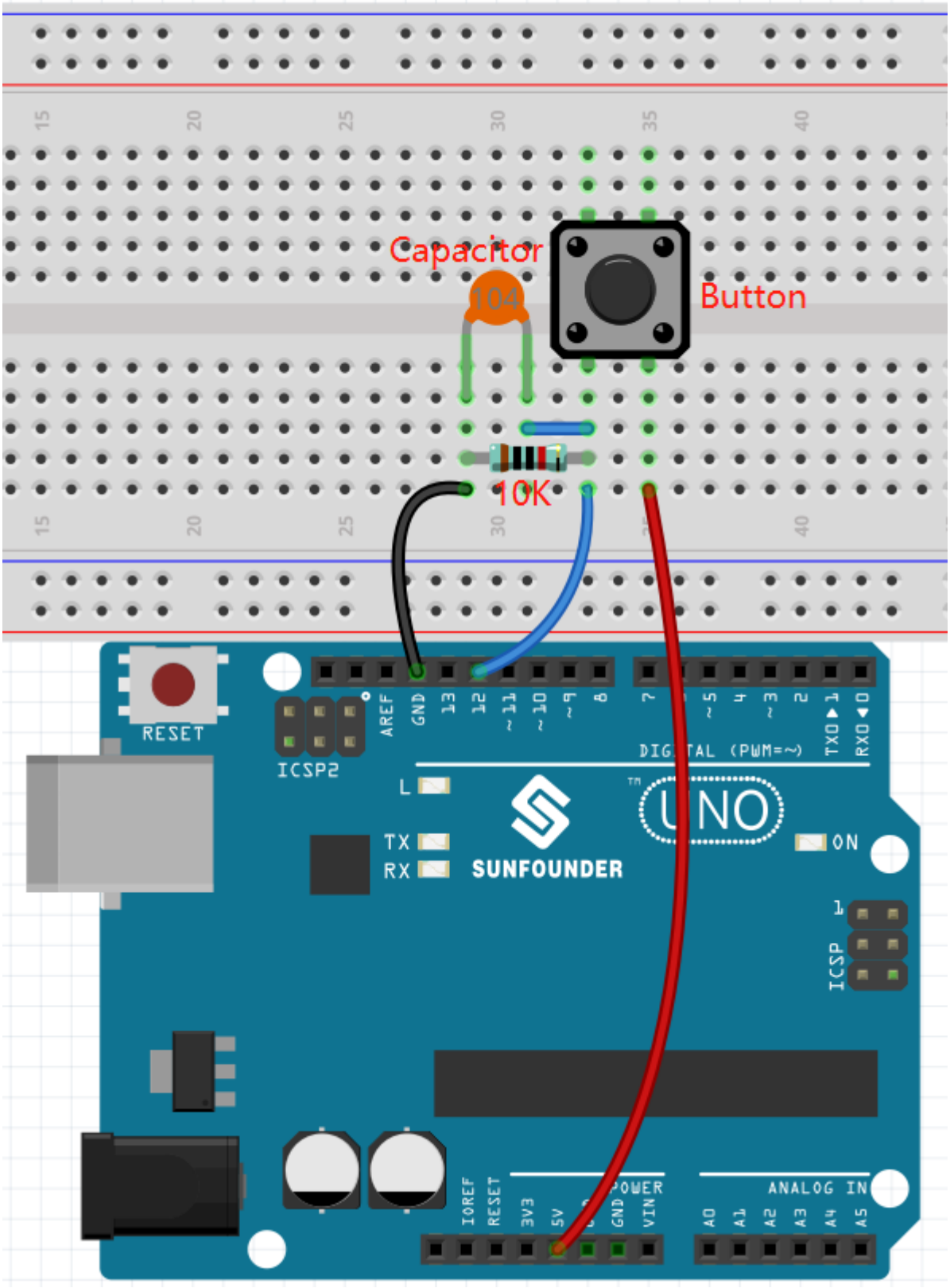
Button



Internal Structure

按照下图搭建电路：

- 将按钮左侧的其中一个引脚连接到 12 引脚，该引脚连接下拉电阻和 0.1uF（104）电容（以消除抖动并在按钮工作时输出稳定电平）。
- 将电阻和电容的另一端连接到 GND，将按钮右侧的一个引脚连接到 5V。


Capacitor
104
Button
10K
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5

- 面包板
- 按键

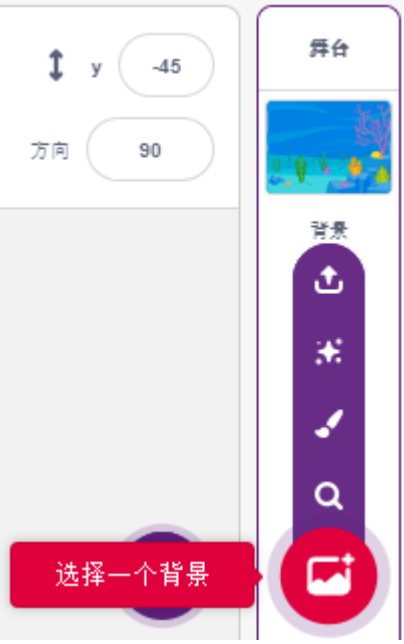
- 电阻
- 电容

编程

我们需要先选择一个 Underwater 的背景，然后添加 Fish 精灵，让它在舞台上来回游动。然后绘制一个 Fishhook 精灵，由按键控制它起钓。当 Fish 精灵碰到 Fishhook 的鱼钩（红色），代表已钓到一条鱼。

1. 添加背景

使用 **选择一个背景**按钮来添加一个 **Underwater** 的背景。

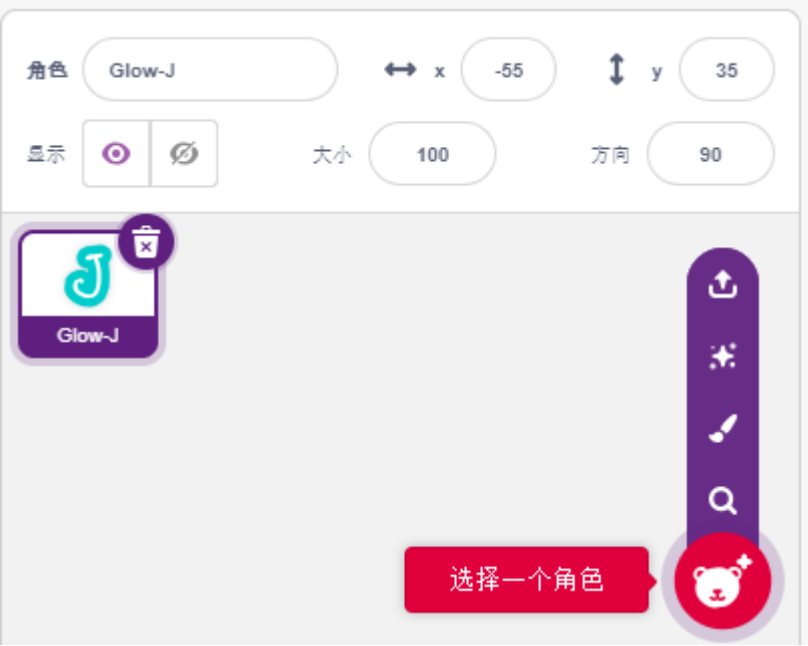


2. Fishhook 精灵

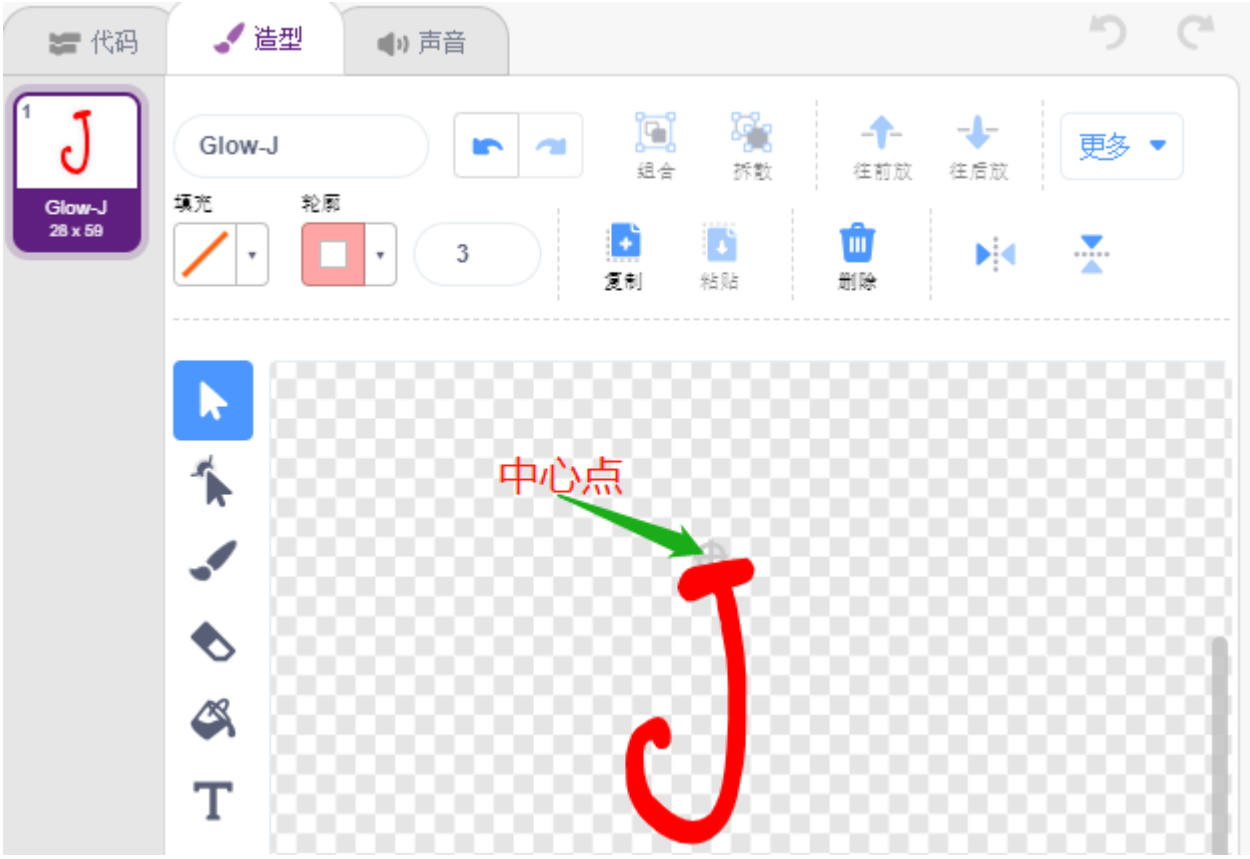
鱼钩精灵要实现的效果是，它平时以黄色状态呆在水底，当按键按下它处于起钓状态（红色），同时它会移动到舞台上方。

在 Pictoblox 中没有鱼钩精灵，我们可以用 Glow-J 精灵修改成鱼钩的样子。

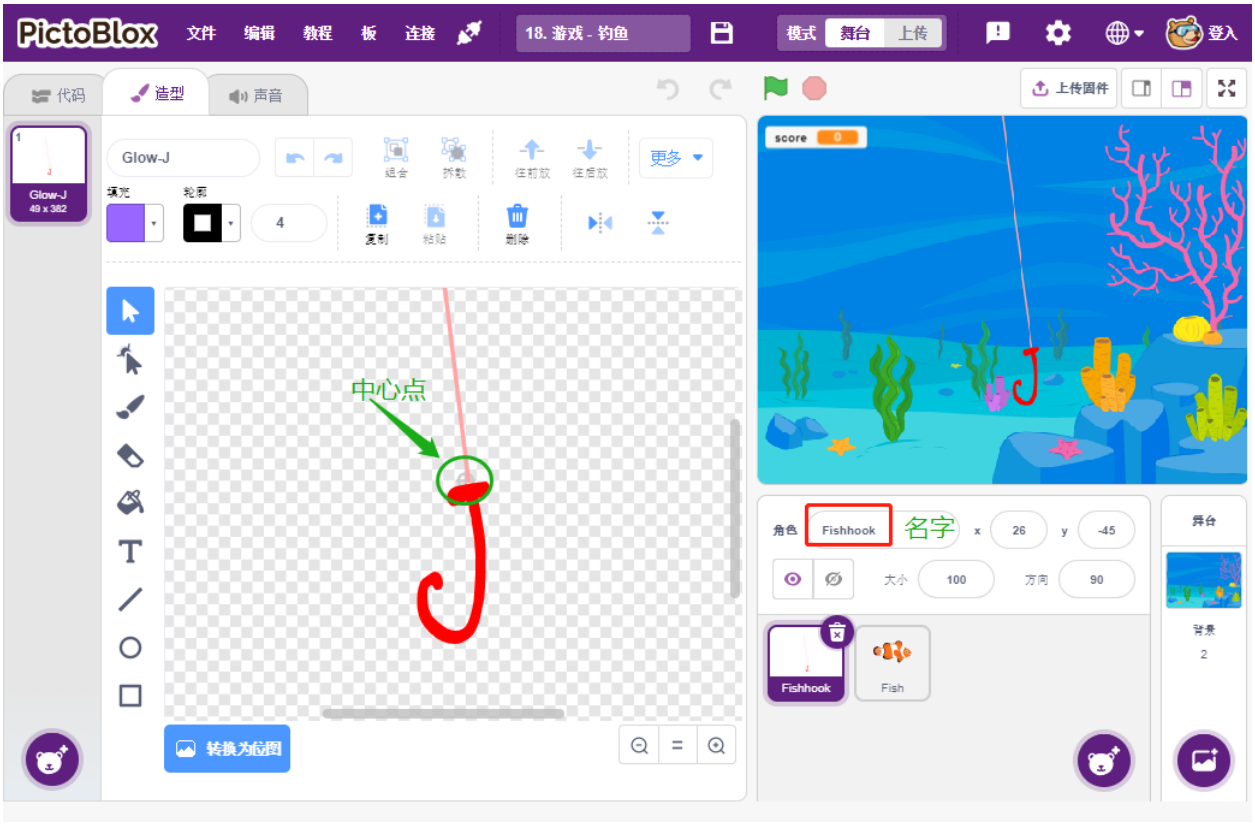
- 通过 **选择一个背景**添加 **Glow-J** 精灵。



- 现在进入到 Glow-J 精灵的 **造型**页面，在画面中选中青色的填充并将它删除。然后将 J 颜色换成红色，可将它的宽度缩小。最需要注意的点是，需要让它的顶部刚好在中心点位置。



- 用 **画线段**工具画一条尽可能长的线，让它在舞台上看起来是从最上面往下放。现在鱼钩精灵已绘制完成，将精灵名字设置为 Fishhook，并将它移动到合适的位置.



- 当绿旗被点击时，设置精灵的颜色效果为 30（yellow），并将它的初始位置设置为（26，-45）。

- 如果按键被按下，则将颜色效果设置为 0（红色，起钓状态），等待 0.1 后，将鱼钩精灵移动到舞台上方。松开按键，让鱼钩回到初始位置。

3. Fish 精灵

fish 精灵要实现的效果是，在舞台上左右移动，当碰到遇到 Fishhook 精灵，则缩小和移动到特定位置后消失，然后再重新克隆 fish 精灵。

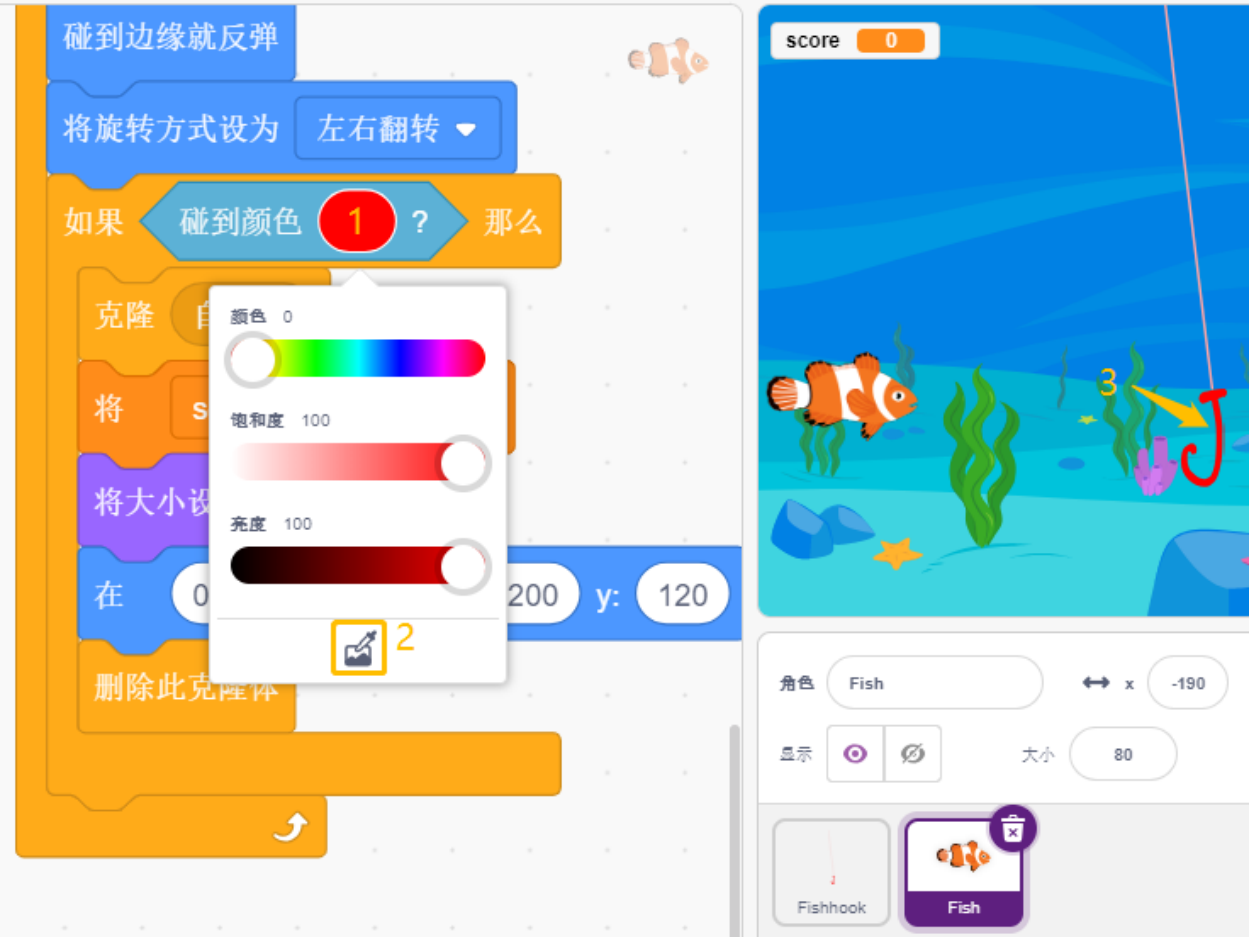
- 添加 **fish** 精灵，并调整它的尺寸和位置。

- 现在给 fish 编写脚本，创建 score 来存放钓到的鱼的数量，将此精灵隐藏并将它克隆。

- 将 fish 精灵的克隆显示，并给它切换造型，设置初始位置为（-190，-50）.

- 让 fish 精灵的克隆，左右移动，碰到边缘则反弹。

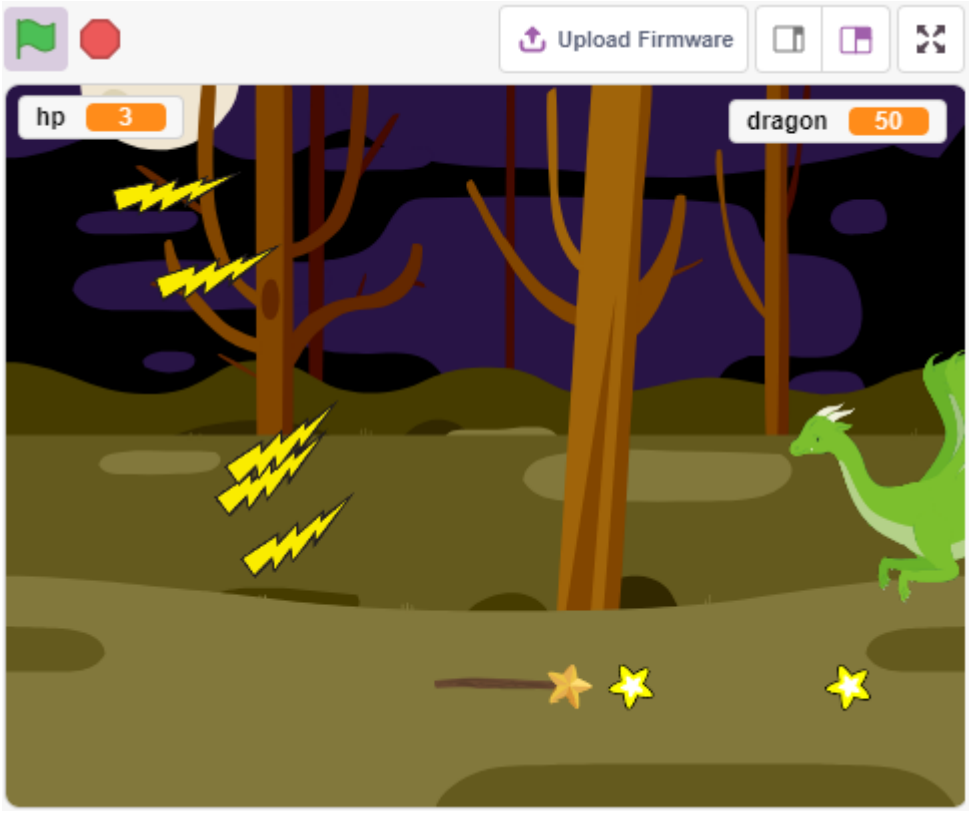
- 鱼的精灵（的克隆）在经过鱼钩精灵时不会有任何反应；当它在钓鱼状态下碰到鱼钩精灵时（变成红色），就会被抓住，这时分数（可变分数）+1，还会出现一个分数动画（缩小 40%，迅速移动到记分牌的位置，然后消失）。同时，一条新的鱼被创造出来（一个新的鱼的克隆精灵），游戏继续。



8.3.19 19. 游戏 - 击败恶龙

在这里，我们使用摇杆来玩 kill drogon 的游戏：

脚本运行后，龙会在右侧上下浮动，并间歇性喷射火焰，你需要通过摇杆控制魔法棒的移动，在避开龙喷射的火焰的同时，向 dragon 发射星星攻击，以将它打败。



搭建电路

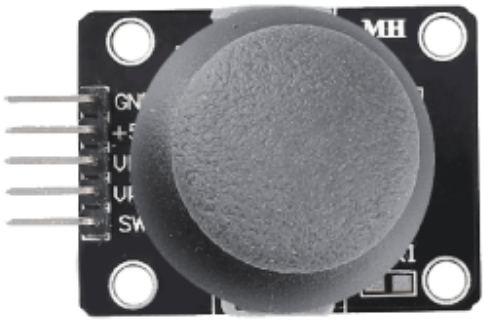
操纵杆是一种输入设备，由一个在底座上旋转的操纵杆组成，并向它所控制的设备报告其角度或方向。操纵杆通常用于控制视频游戏和机器人。

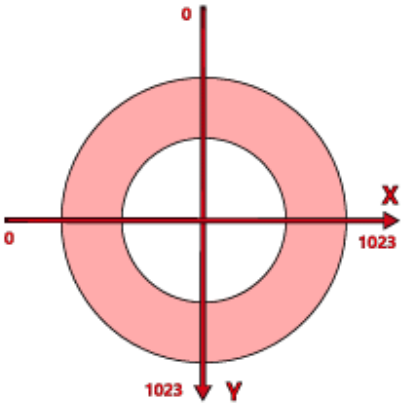
为了将完整的运动范围传达给计算机，操纵杆需要在两个轴上测量操纵杆的位置——X 轴（从左到右）和 Y 轴（上下）。

摇杆的运动坐标如下图所示：

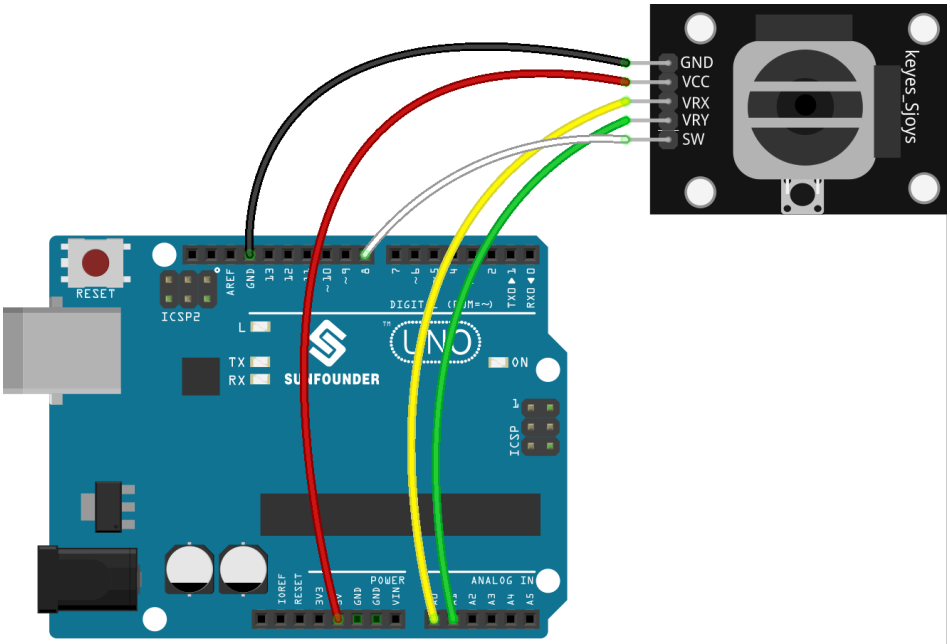
备注：

- x 坐标是从左到有，范围是 0-1023
- y 坐标是从下到上，范围是 0-1023





现在按照下图搭建电路.

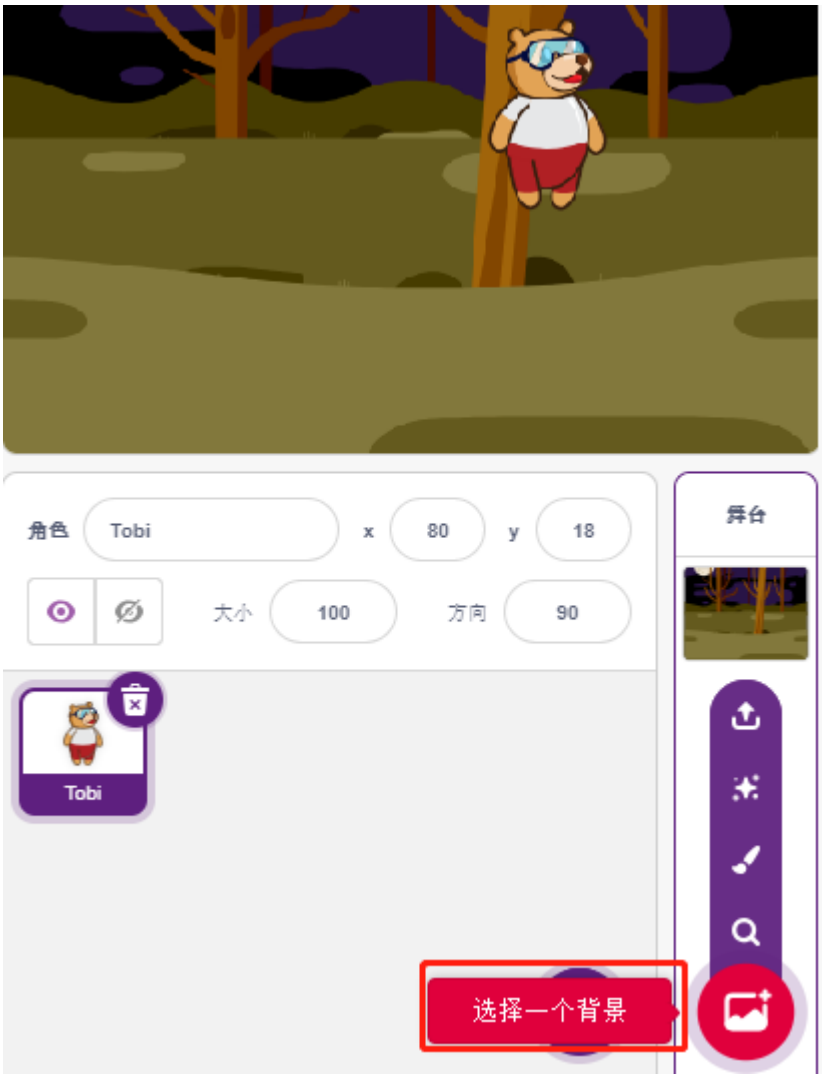


- 面包板
- 摇杆模块

编程

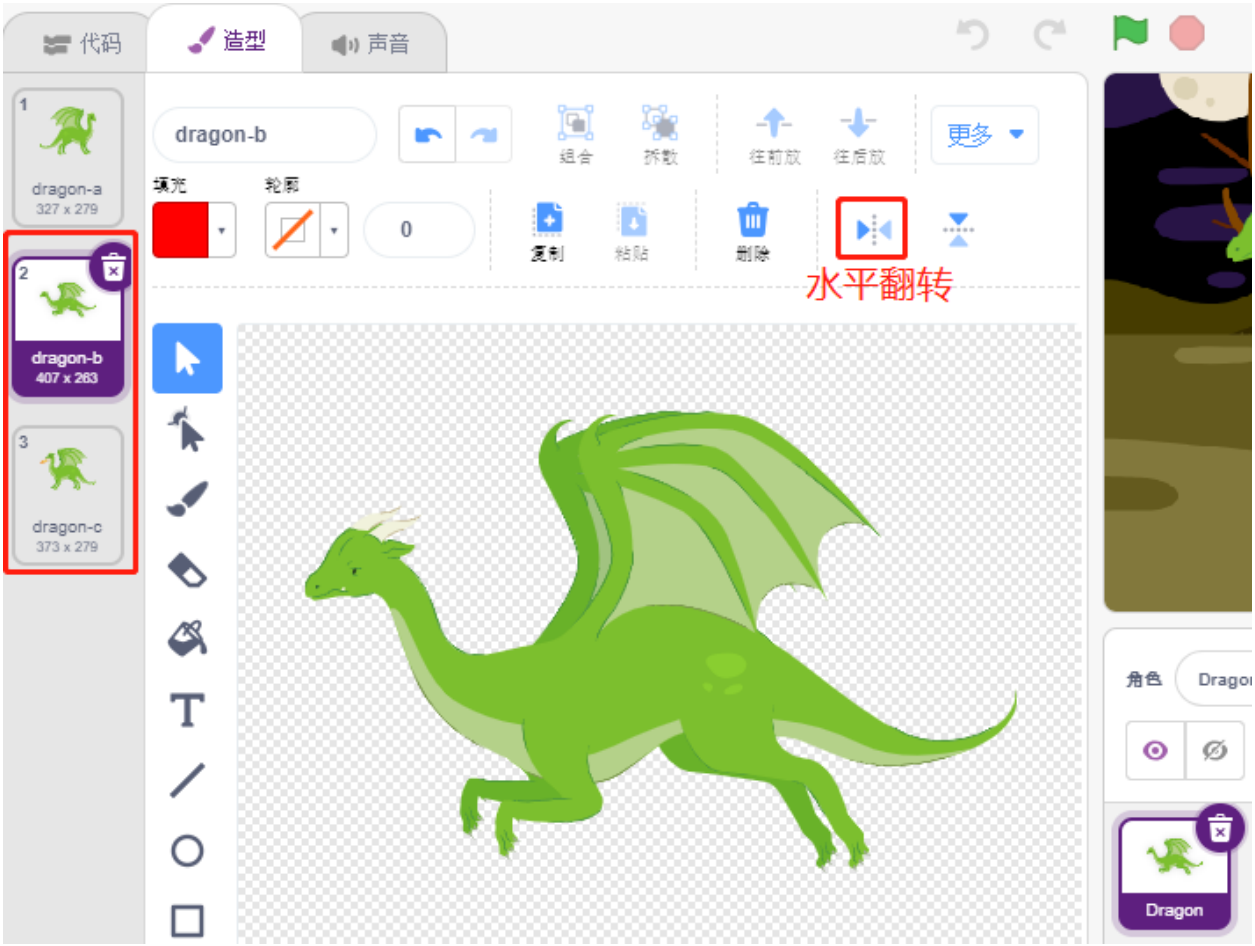
1. Dragon

- 通过 **选择一个背景**按键添加 **Woods** 背景。



- 删除原有的精灵，添加 **Dragon** 精灵。

- 进入 **造型**页面，将 **dragon-b** 和 **dragon-c** 造型水平翻转。



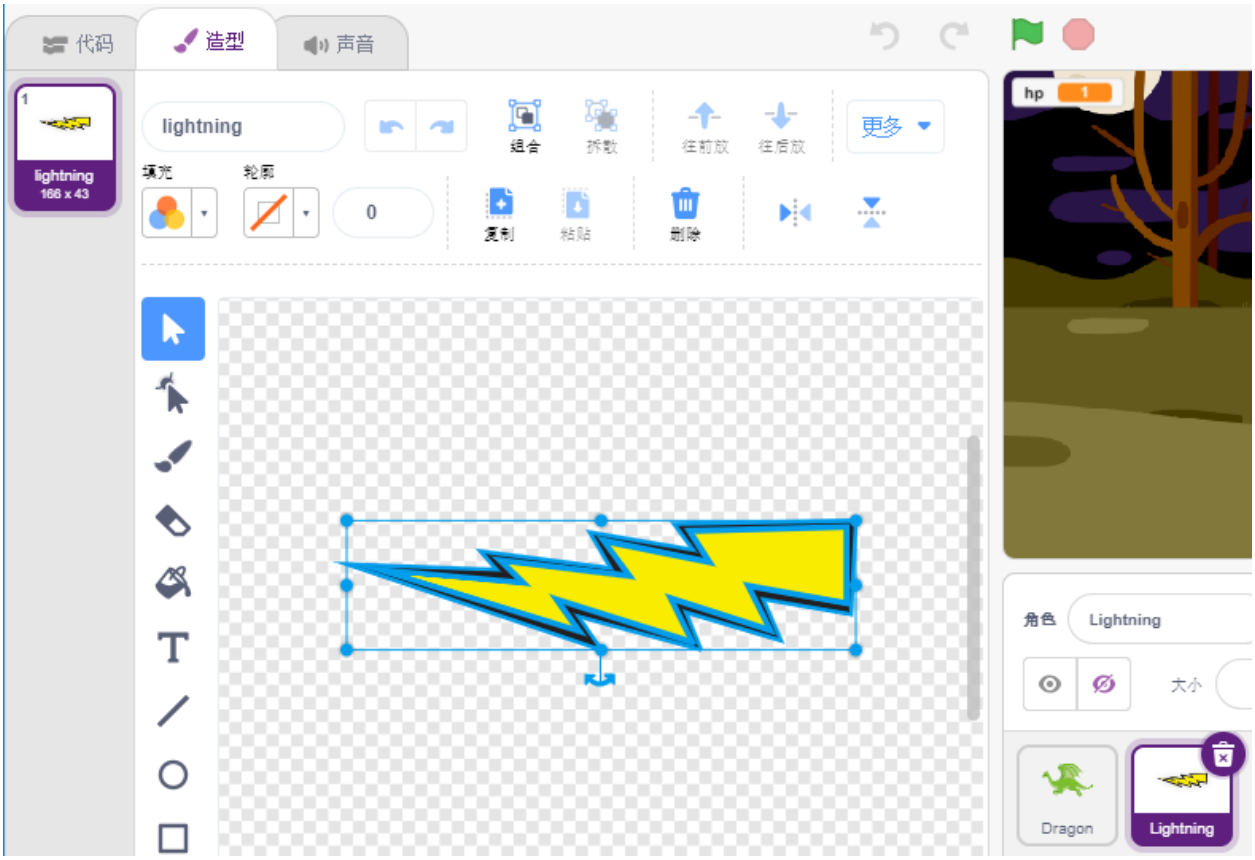
- 将尺寸设置为 50%。

- 现在开始为 Dragon 精灵编写脚本，先创建变量 dragon 来记录龙的生命值，初始值设置为 50。

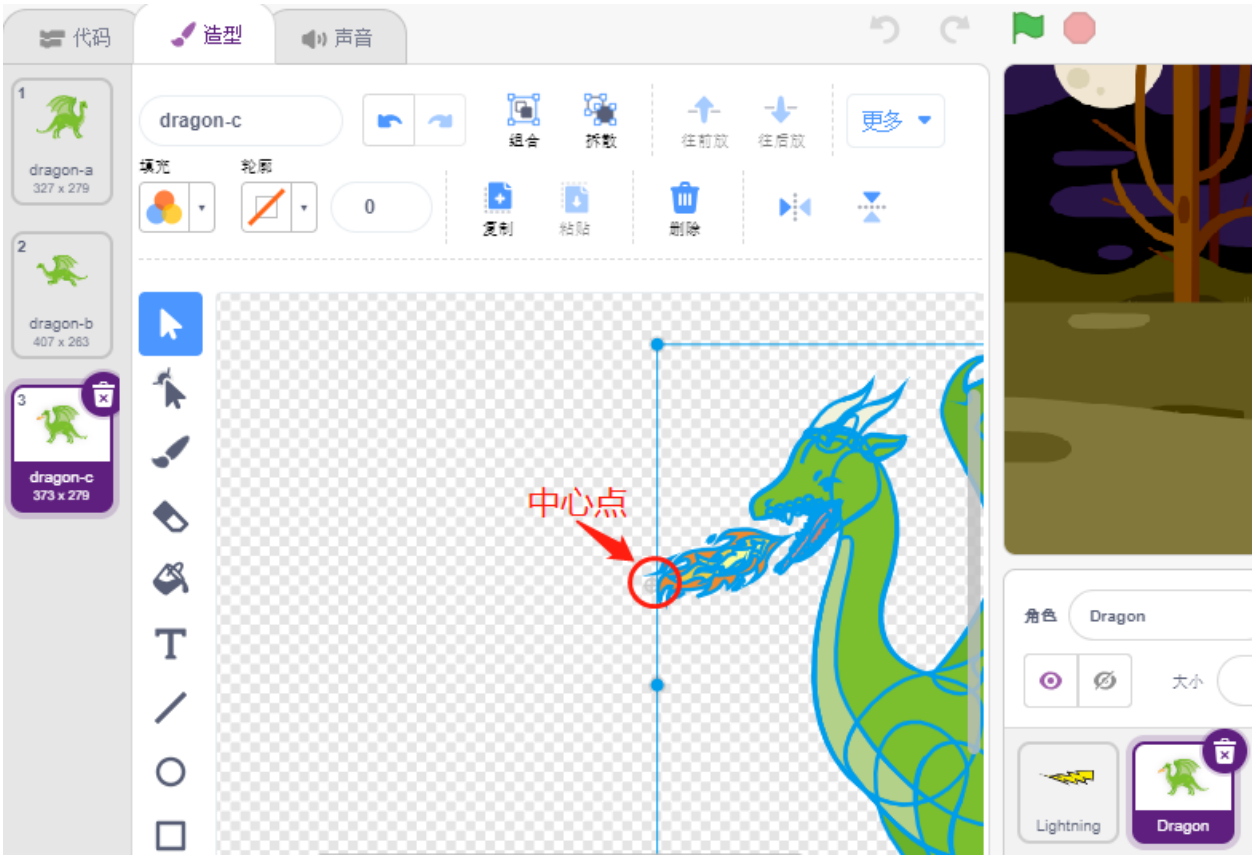
- 接下来将精灵造型切换为 dragon-b，并让 Dragon 精灵在一个范围内上下徘徊。

- 添加一个 **Lightning** 精灵作为巨龙喷出的火焰。你需要在造型中将它顺时针转 90°，这是为了让闪电发射时的移动方向正确。

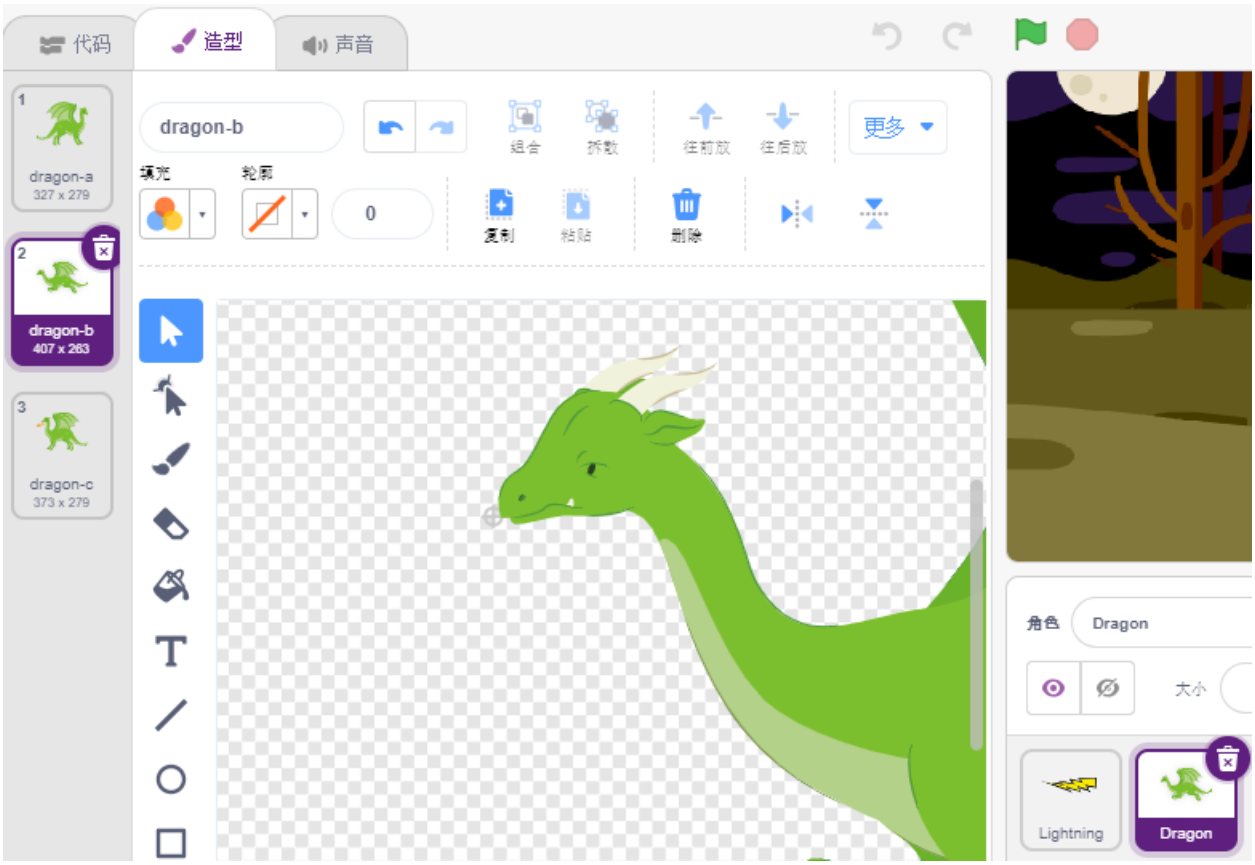
备注： 在调整 Lightning 精灵的造型时，你可能会将它移的偏离中心点，这是必须避免的！中心点必须在精灵正中央的位置！



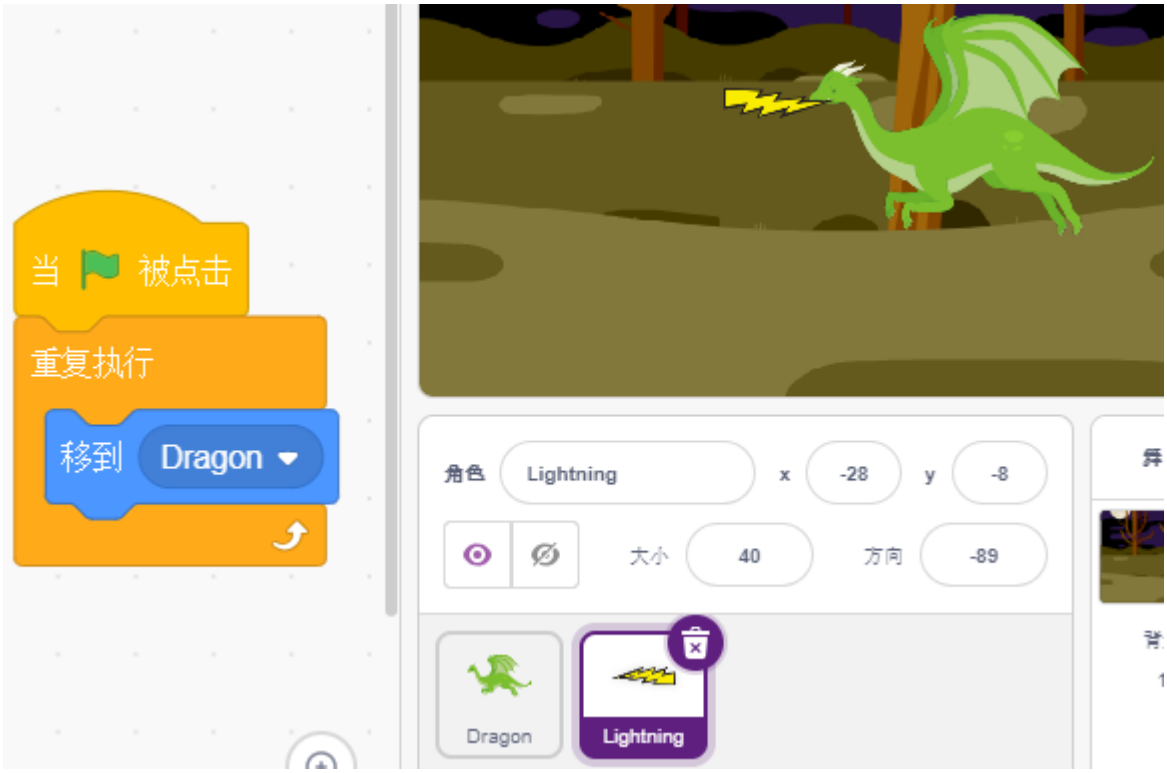
- 随后你需要调整 Dragon 精灵的 dragon-c 造型，中心点应当位于巨龙口中的火焰处。这样做是为了让 dragon 精灵和和 Lightning 精灵的位置能对应的上，避免闪电从巨龙的脚底发射出来。



- 相应的，dragon-b 也需要调整位置，让龙头和中心点重合。



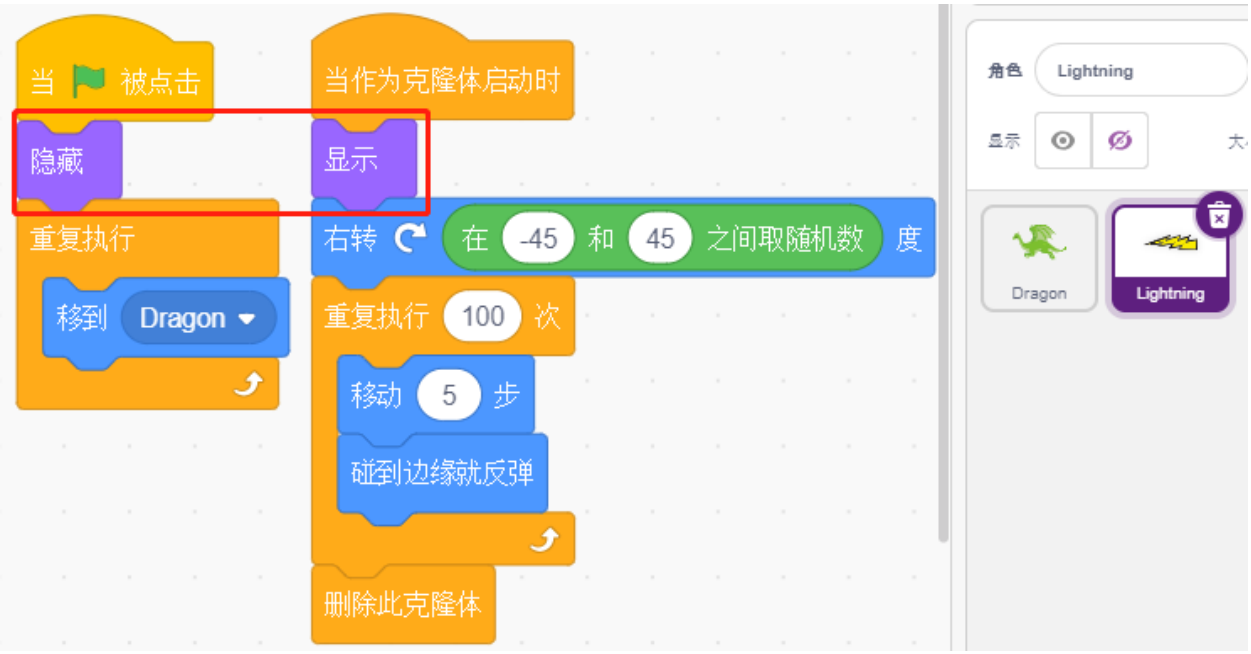
- 调整 Lightning 精灵的大小和方向，让画面看起来更和谐。现在为 Lightning 精灵的本体编写脚本，这很容易，只需要让它永远跟着 Dragon 精灵就可以了。此时点击绿旗，你将看到巨龙衔着闪电移动的情景。



- 回到 Dragon 精灵，现在让它喷出闪电，注意，不是让口中衔着的闪电射出去，而是为 Lightning 精灵创造克隆体。

- 打开 Lightning 精灵，为 Lightning 克隆体添加功能，随机调整角度后射出，碰撞墙壁会反弹，并在一定时间后消失。

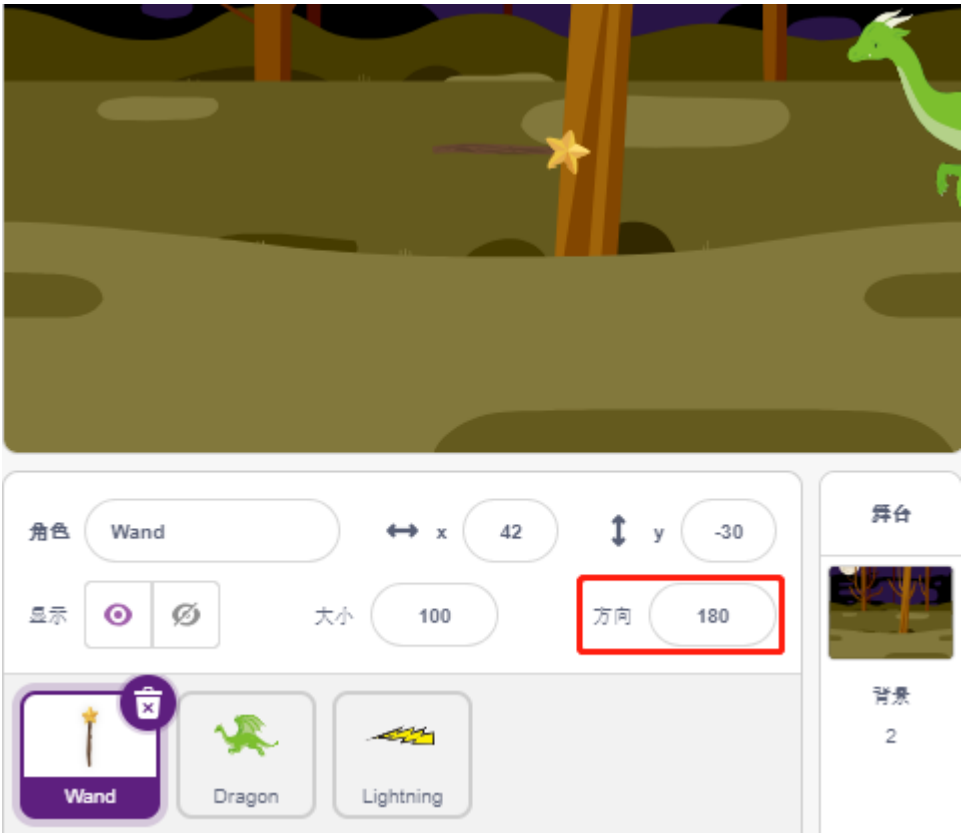
- 在 Lightning 精灵中，隐藏它的本体，显示克隆体。



现在，巨龙可以徘徊并喷射闪电了！

2.Wand

- 创建一个 **Wand 精灵**，将其方向旋转至 180，即指向右侧。



- 为 Wand 精灵编写脚本，先创建变量来记录它的生命值，初始值设置为 3. 随后读取 Joystick 的值，用来控制魔杖的移动。

- 巨龙拥有闪电，讨伐巨龙的魔杖也拥有它的“魔法子弹”！创建一个 **Star** 精灵，调整它的大小，并编写脚本使其永远跟随 Wand 精灵，并且，限制星星的数量为三颗。

- 让魔杖自动射出星星，魔杖发射星星的方式与巨龙发射雷电的方式一样——创建克隆体。

- 回到 Star 精灵，为它的克隆体编写脚本，令它旋转着往右射出，在超出舞台后消失并恢复星星个数。跟闪电一样，隐藏本体，显示克隆体。

现在，我们拥有一根会射出星星子弹的魔杖了。

3. Fight!

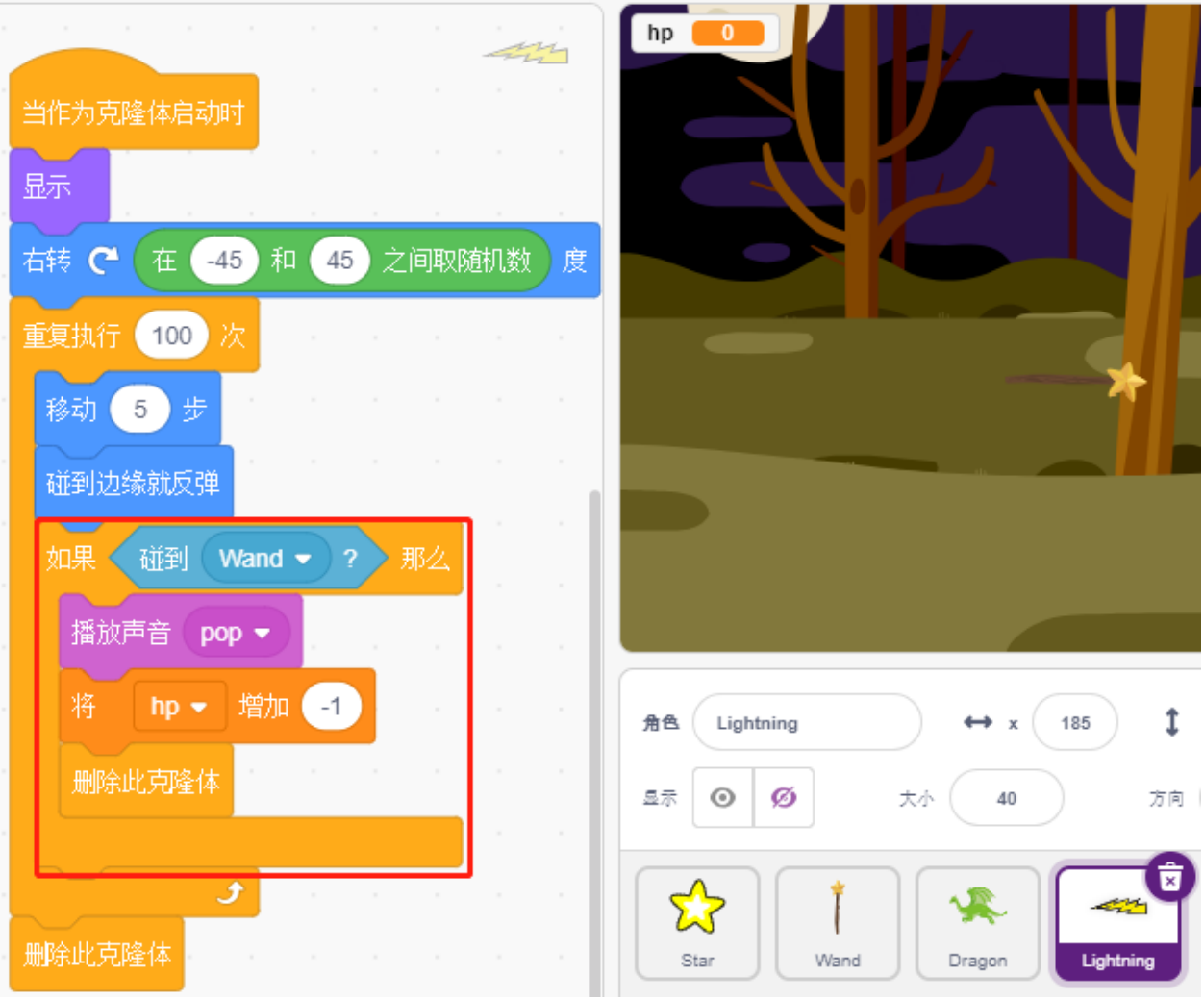
魔杖和巨龙目前还是各玩各的，未能影响对方，我们要让它们战斗起来。巨龙很强壮，魔杖则是讨伐巨龙的勇者，它们之间的交互包括了以下部分：

1. 如果魔杖碰到巨龙，会被击退并且损失生命值。
2. 如果闪电击中魔杖，魔杖会损失生命值。
3. 如果星星子弹击中巨龙，巨龙则会损失生命值。

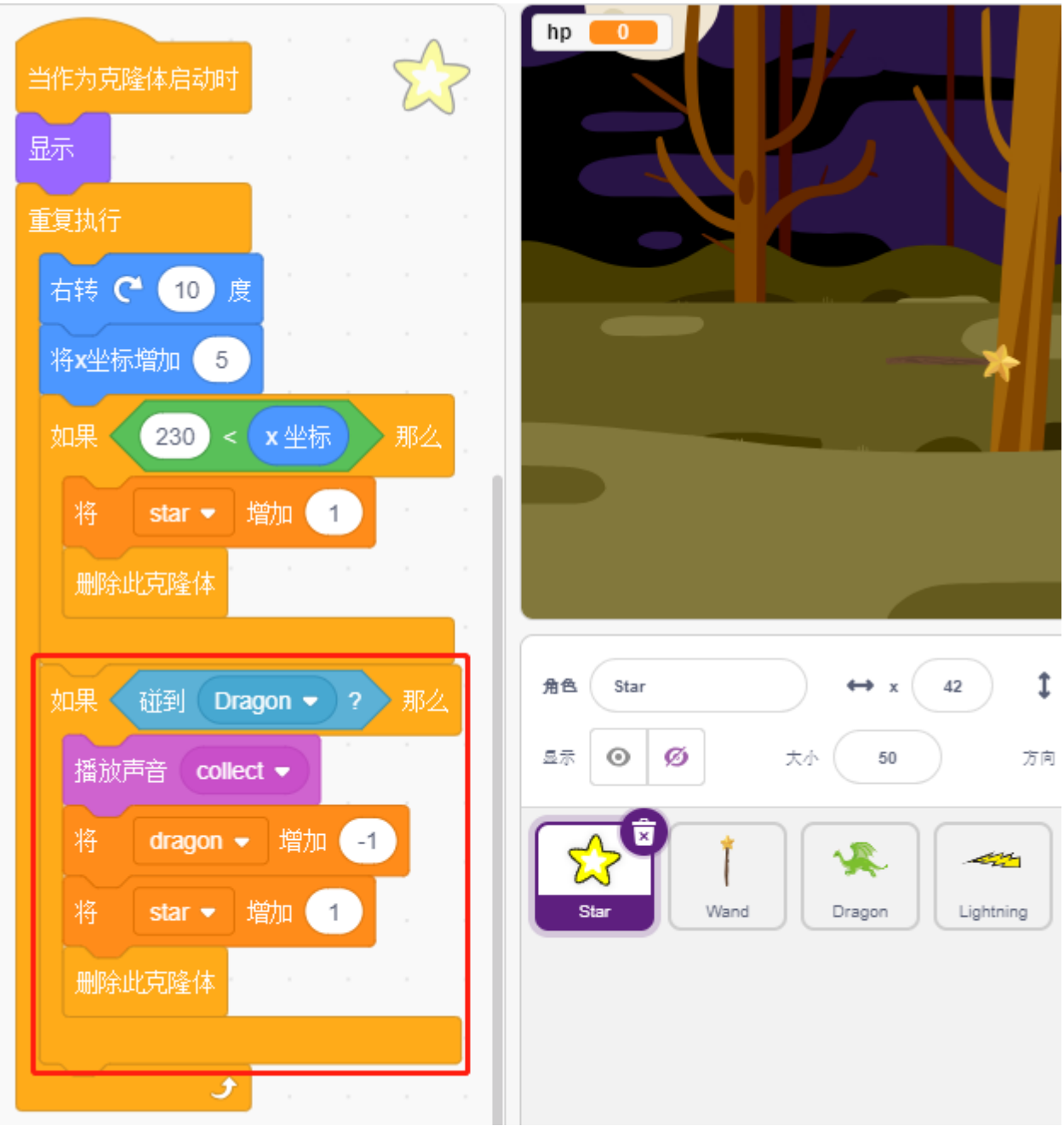
梳理清楚后，我们来继续改动各个精灵的脚本。

- 如果魔杖碰到巨龙，会被击退并且损失生命值。

- 如果闪电（Lightning 精灵的克隆体）击中魔杖，它会发出 pop 音效并消失，魔杖则是会损失生命值。



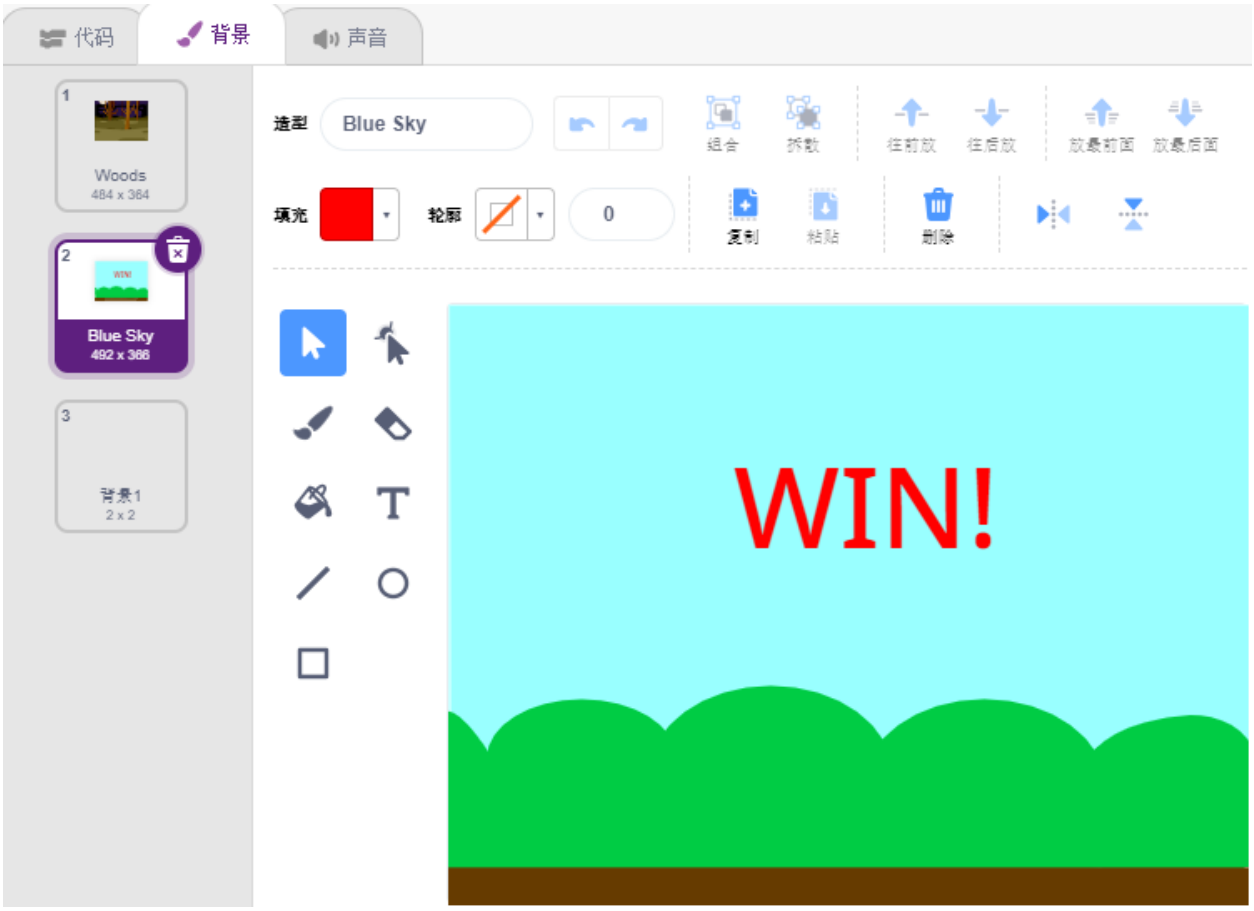
- 如果星星子弹（Star 精灵的克隆体）击中巨龙，它会发出 collect 音效并消失，同时恢复星星子弹的计数，巨龙则是会损失生命值。



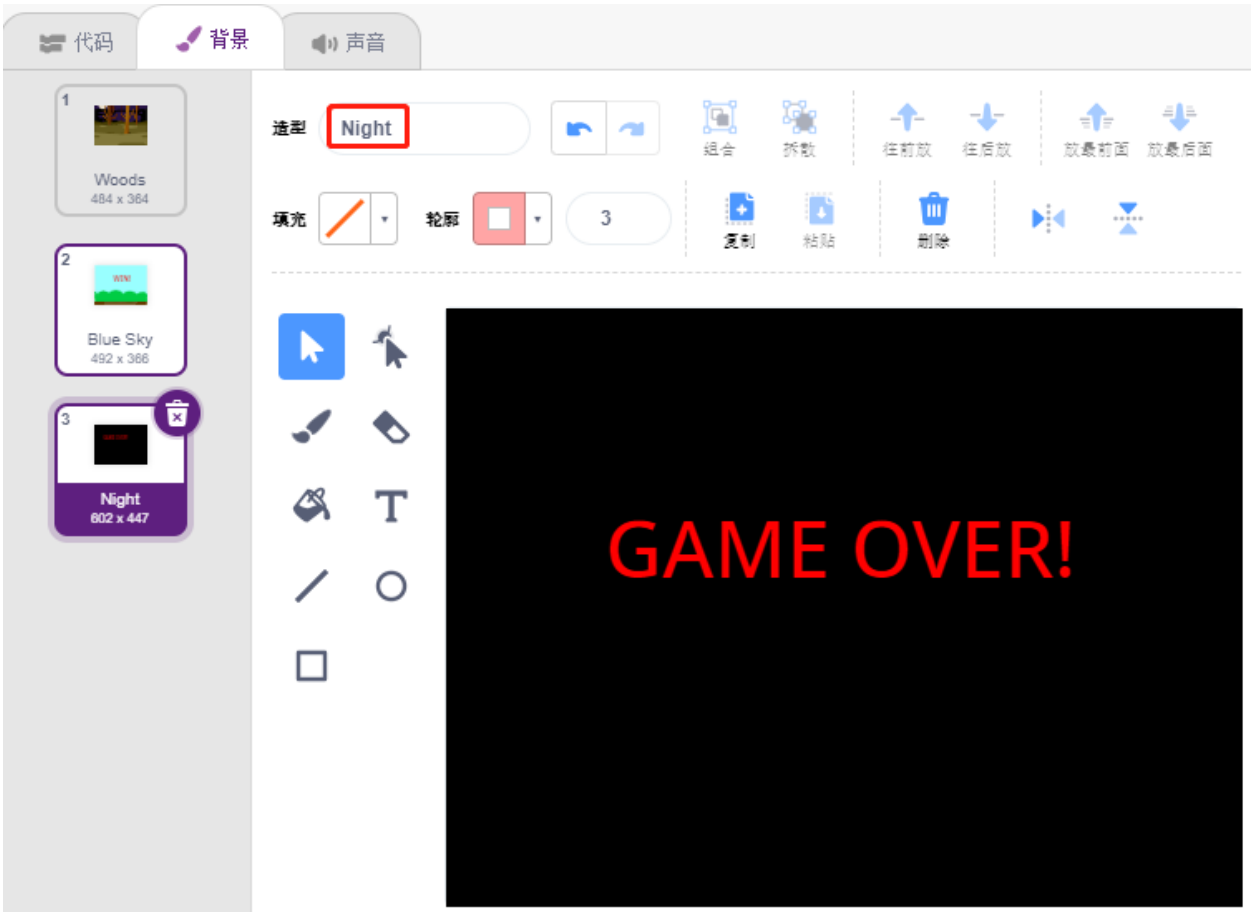
4. 舞台

魔杖与巨龙的战斗最终将分出胜负，我们用舞台来表示。

- 添加 **Blue Sky** 背景, 并在 Blue Sky 背景上写上字符“WIN!”来代表恶龙被杀死，黎明到来。



- 并将空白背景修改如下，用来代表游戏失败，一切都将陷入黑暗。



- 现在编写脚本来切换这些背景，当绿旗被点击时，切换到 Woods 背景；如果恶龙的生命值小于 1 ，则游戏成功，将背景切换到 Blue Sky；如果你的生命值小于 1，则切换到 Night 背景，游戏失败。


当 被点击
换成 Woods 背景
重复执行
如果 dragon < 1 那么
换成 Blue Sky 2 背景
停止 全部脚本
如果 hp < 1 那么
换成 Night 背景
停止 全部脚本

CHAPTER 9

常见问题解答

9.1 1、板子不工作？

如果您在 PictoBlox 的舞台模式下单击了绿色标志或精灵，则无法向 Arduino 板发送数据或从 Arduino 板接收数据。

您需要执行以下操作。

- 检查是否选择了正确的板并已连接。
- 检查是否选择了正确的引脚。
- 如果您是第一次在 PictoBlox 中使用此 Arduino 板，或者此 Arduino 板之前已使用 Arduino IDE 上传代码。然后你需要在使用之前点击上传固件。

9.2 2、COMxx”：访问被拒绝？

在 Arduino IDE 中上传代码时，它显示: `avrdude: ser_open(): can't open device “。COMxx”: 访问被拒绝。`

- 检查您是否选择了正确的 COM 端口。
- 如果您打开 PictoBlox 并且同时连接了开发板，您需要在 PictoBlox 中断开它并在 Arduino IDE 上重新上传它。

9.3 3、如何在 PictoBlox 的舞台模式下工作？

详细教程请参考舞台模式。

9.4 4、如何在 PictoBlox 的上传模式下工作？

详细教程请参考上传模式。

9.5 5、在 PictoBlox 中编译失败？

- 检查是否有选择正确的板和端口
- 检查代码是否正确
- 如果都没有问题，建议重启 PictoBlox。

CHAPTER 10

版权声明