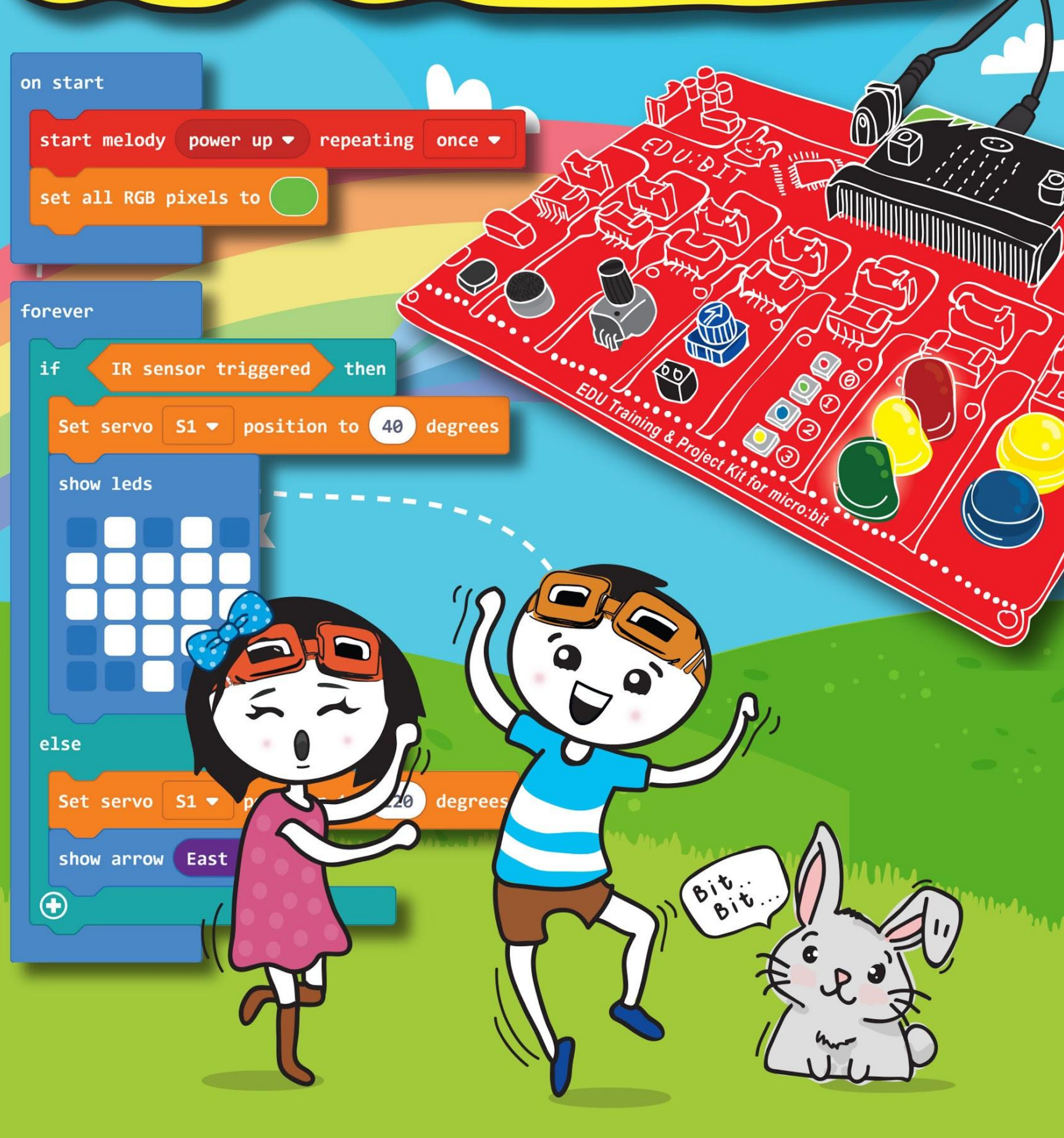
EDU:BIT书
(中文翻译)

# Explore STEM & Coding with EDU:BIT



# RERO EDUTEAM @ CYTRON 的笔记

致 ________________ ，
（孩子的名字）

你有听说过 micro:bit 吗？Micro:bit 是一款由英国设计的微型电脑开发板并已推广到全世界，从而驱动青少年学习编程。

Cytron 的工程师通过构建此 EDU:BIT 将其提升到了一个新的水平，以便你可以一点一点地逐步学习编码。EDU:BIT 具有带压电蜂鸣器和音频插孔的 Music Bit，可播放音乐，Sound Bit 用于检测噪声，Potentio Bit 用于模拟控制，IR Bit 用于检测物体，RGB Bit 用于彩色照明显示，Traffic Light Bit 带有红色，黄色和绿色 LED，Button Bit 则是在 micro:bit 上的按钮放大版本。该套软件还具有 DC Motor 和 Servo Motor。 多么酷啊？

我们开始吧！ 在接下来的页面中，我们将探索和重新创建一些经典的童年游戏，例如 剪刀石头布， 蛇和梯子， 捉人游戏， 才艺表演，“Twister"，"Simon说”和其他有趣的游戏。按照一步一步的指导，建立游戏，然后与你的朋友分享玩的乐趣！ 随意修改代码以创建自己的游戏升级版本。

在每章的结尾，都会有一个挑战，要求你使用自己的知识来构建课堂应用程序。 试试看，如果你遇到困难，我们将始终在这里为你提供帮助。

准备好了吗？ 让我们开始这个令人振奋的征途
- 有乐趣的学习和探索！

真诚的，
亚当 和 安娜

# 使用 EDU:Bit 培训和项目工具包探索STEM和编码

**撰写者**
Cheryl Ng, SC Lim & Adrian Teo

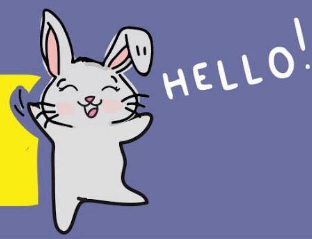
**绘图者**
Suhana Oazmi

**中文翻译团队**
Jeffrey Liu, Tan Ling Zhi, Chew Soo Lee

2020

出版商 Cytron Technologies

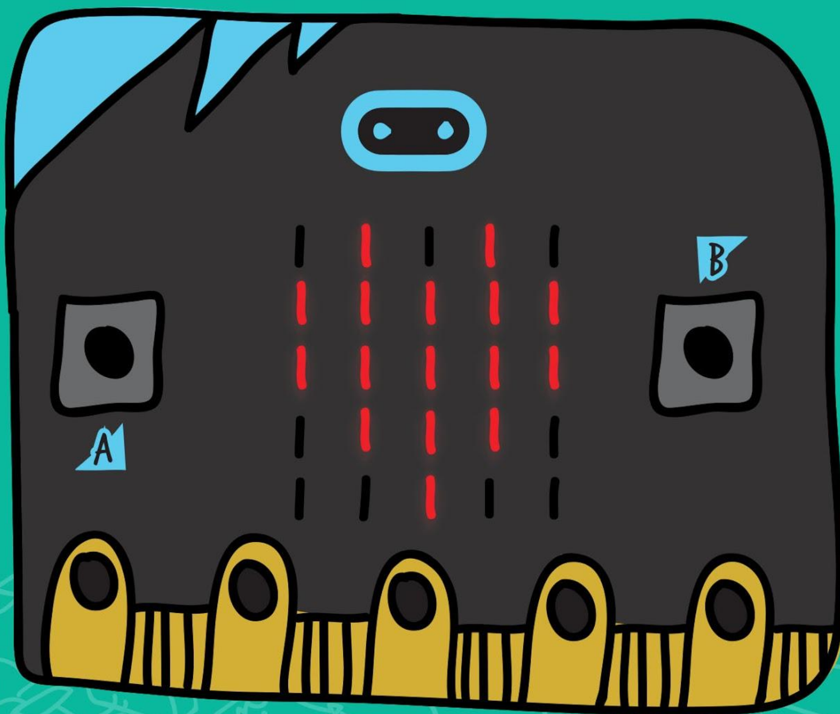
# 目录

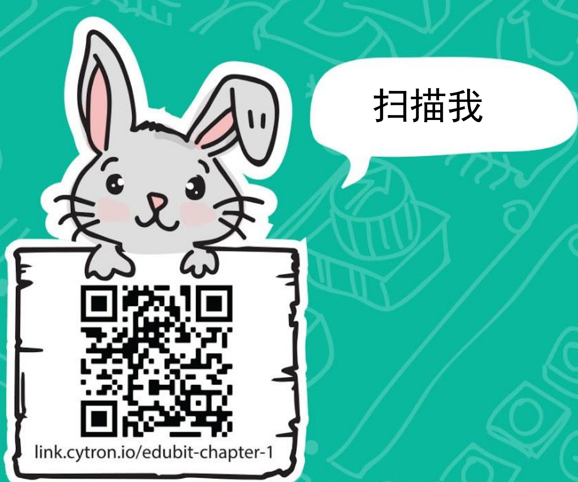


第1章

# > 你好，世界!

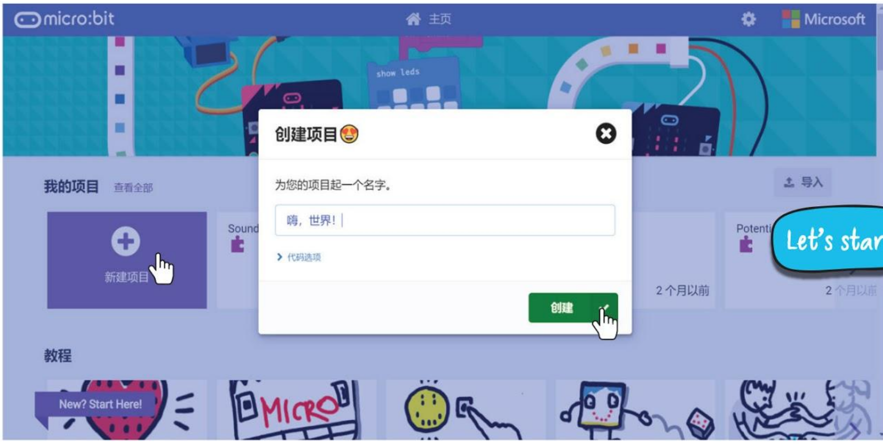
micro:bit上的LED矩阵





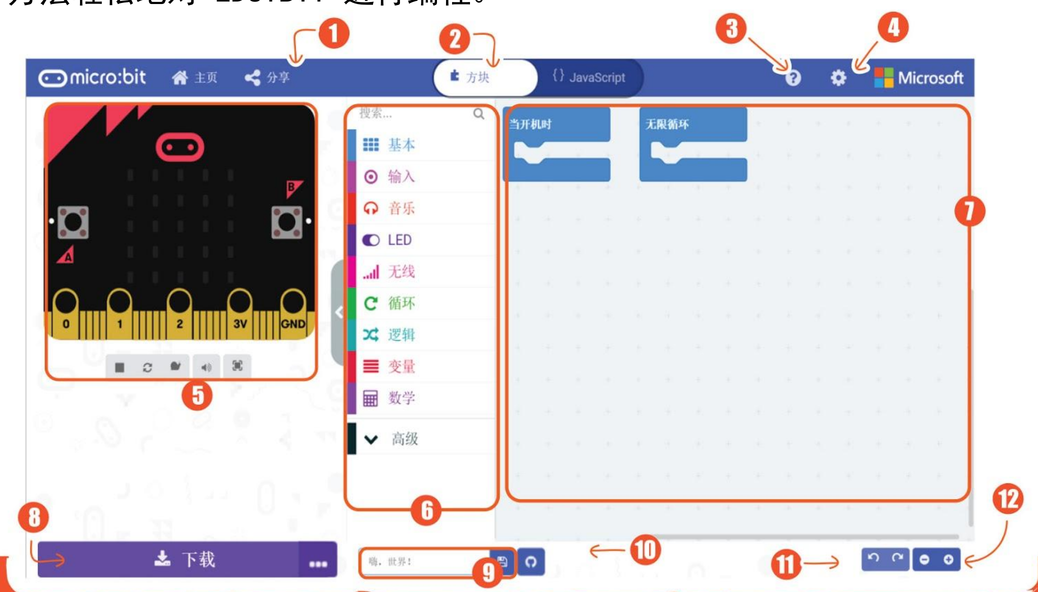
## 让我们编码！

第1步 打开浏览器，输入 https://makecode.microbit.org . 点击 “**新建项目**” 输入你的项目名称，然后单击“**创建**”。



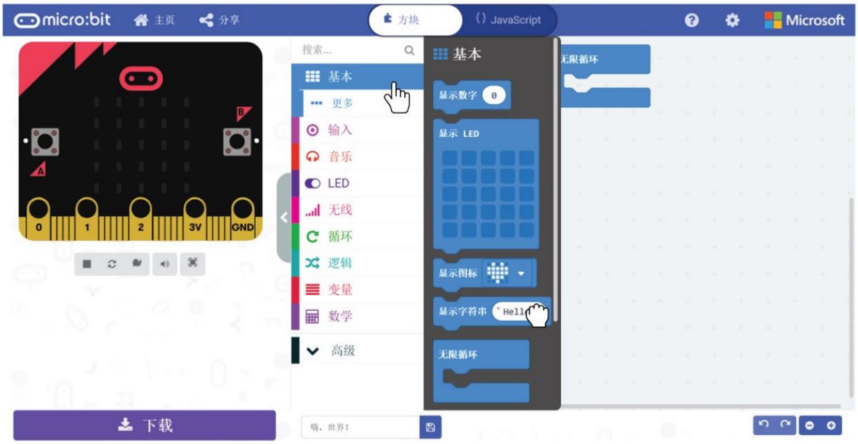


你将看到此 Microsoft MakeCode 编辑器页面，该页面使你可以使用拖放方法轻松地对 EDU:BIT 进行编程。

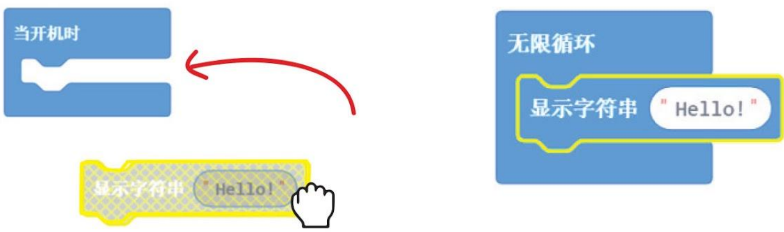



1）发布和共享你的项目
2）选择使用方块，JavaScript 或 Python 进行编程。
3）打开帮助画面
4）更改设置，添加扩展并配对设备。
5）**模拟器** – 显示你的程序在 micro:bit 上运行的样子
6）**工具箱/分类抽屉** – 单击以查看每个类别中的可用编码块。 同一类别的所有块均具有相同的颜色。
7） **编程工作区** – 通过在该区域将程序块对齐在一起来构造程序。
8） 将你的代码下载到 micro:bit
9） 输入名称并将当前项目保存到你的计算机
10）创建GitHub资料库
11）撤销重做
12）放大/缩小

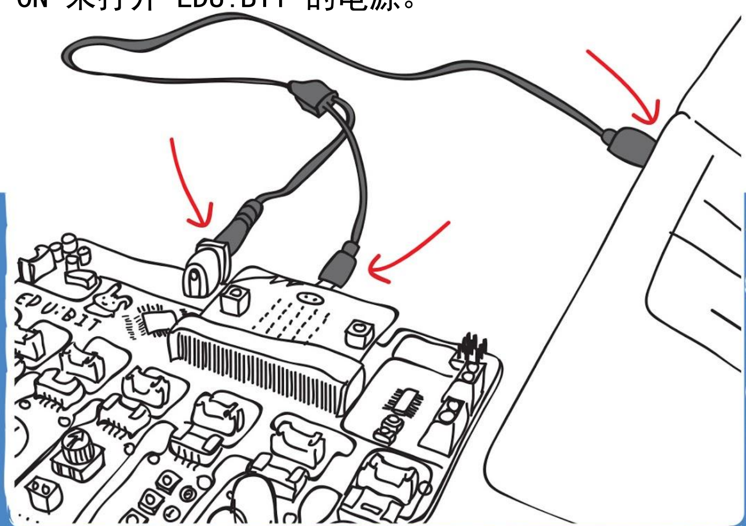
第2步 单击［基本］，然后选择［显示字符串］块。



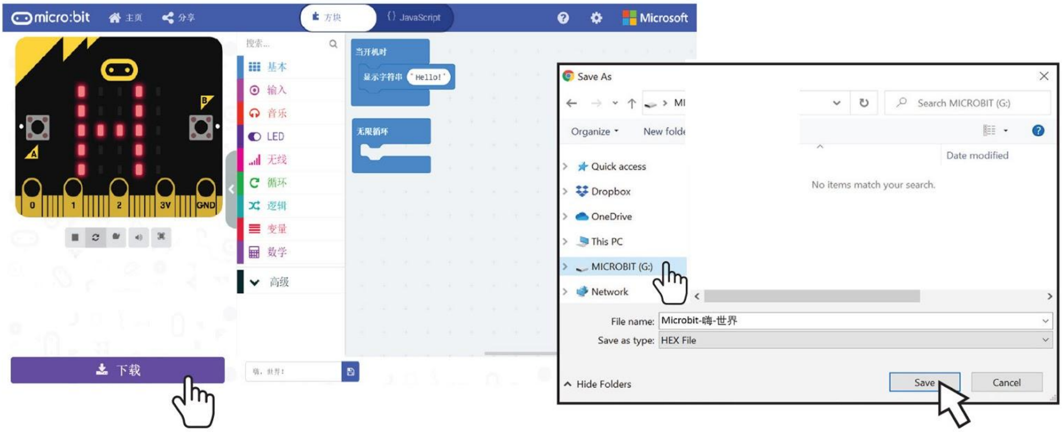
第3步 单击并将［显示字符串］块对齐到［当开机时］插槽。




第4步 如图所示，将 USB 电缆连接到计算机和 EDU:BIT。 切记通过将开关滑动到 ON 来打开 EDU:BIT 的电源。



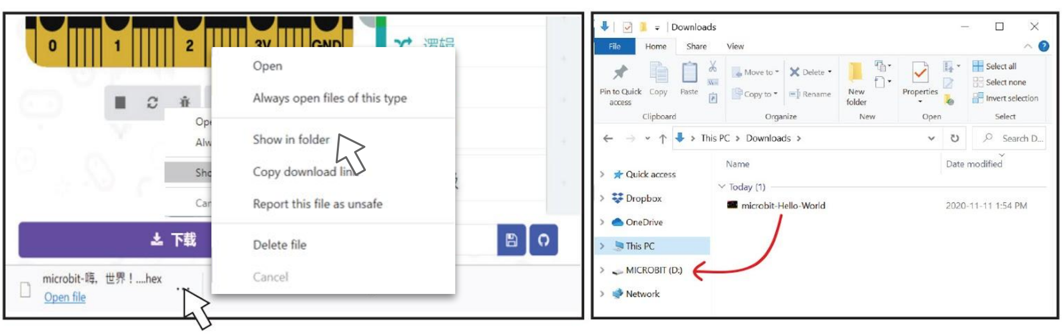
第5步 点击［下载］按钮。 在弹出窗口中，选择将项目下载到 MICROBIT 驱动器。 显示“下载完毕”时，关闭窗口。



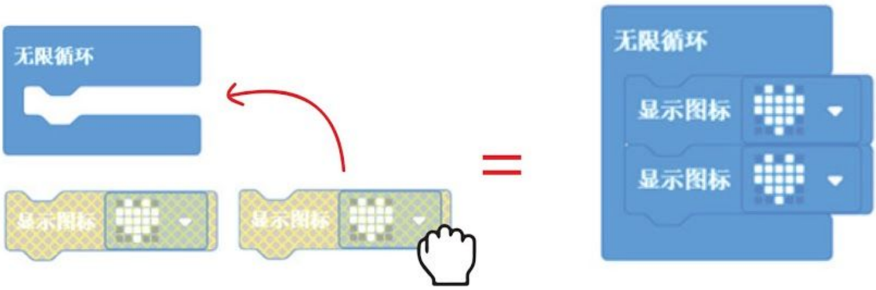

传输代码的过程称为刷新。 传输过程中，micro:bit 背面的橙色 LED 闪烁，完成后，代码将自动运行。

## 注意

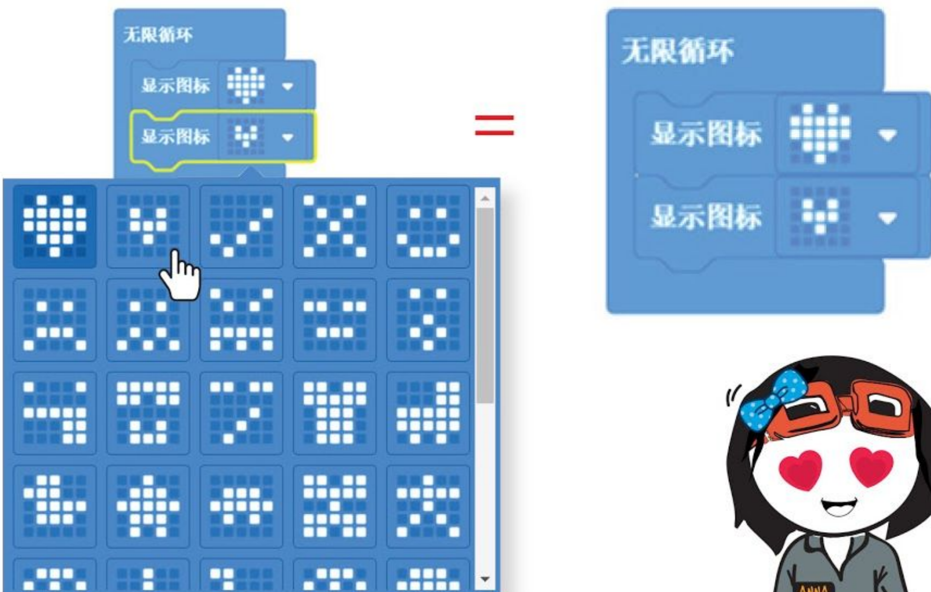
如果没有出现弹出窗口，则表示文件已自动下载到浏览器设置为保存下载的位置。 右键单击将出现在窗口底部的下载的 .hex 文件，然后选择“Show in folder”。 单击下载的“ microbit-xxxx.hex”文件并将其拖动到MICROBIT驱动器，就像你将文件复制到闪存驱动器一样。



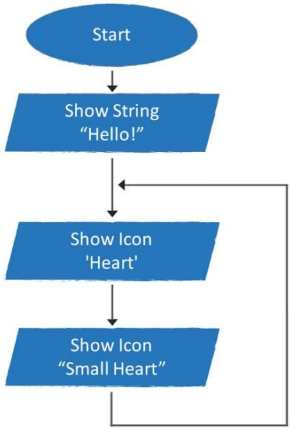
第6步 单击［基本］，然后单击［显示图标］块。 重复添加另一个［显示图标］块。 单击并将［显示图标］块锁定到［无限循环］插槽。




第7步 左键单击第二个［显示图标］块的图标，然后从弹出窗口中选择“small heart”设计。 将代码刷新到你的 EDU:BIT。

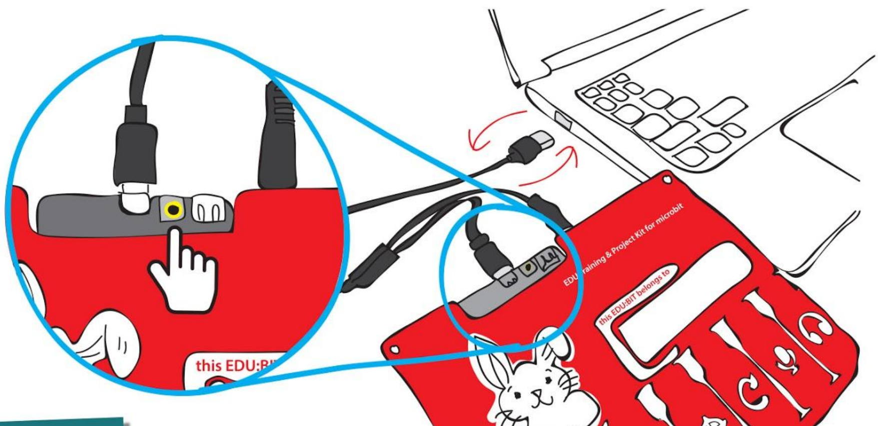



# 破解密码



**[当开机时]** 块在启动时运行一次代码。
**[无限循环]** **程序**块一次又一次地（永远）运行代码。

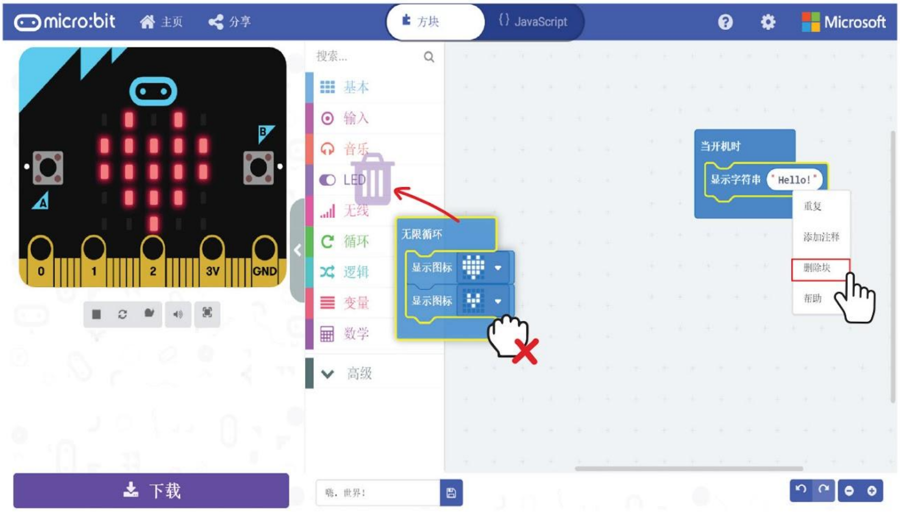


## 注意

如果你想重新启动该程序，只需按 RESET 按钮或拔下 USB 电缆，然后再将其重新插入即可复位。

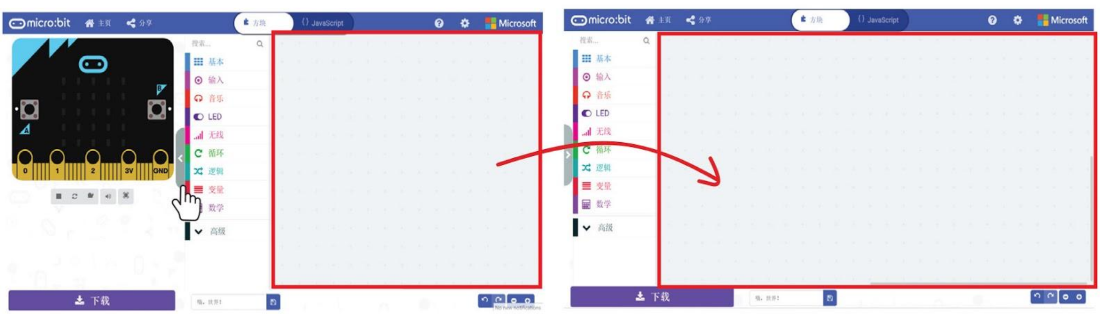
## 快速提示#1！

你可以通过单击不需要的块并将其拖到“工具箱/分类抽屉”区域中来删除它们。 出现垃圾桶图标时，松开以删除该块。 或者，你可以右键单击该块，然后选择“删除块”。



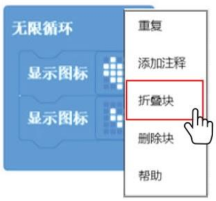

## 快速提示#2！

如果不使用模拟器窗口，则可以单击选项卡将其隐藏，以便为编码块留出更多空间。



## 快速提示#3！

你可以通过右键单击块群来折叠块群，然后选择“折叠块”。 要展开一个折叠的块簇，只需单击(V)图标。





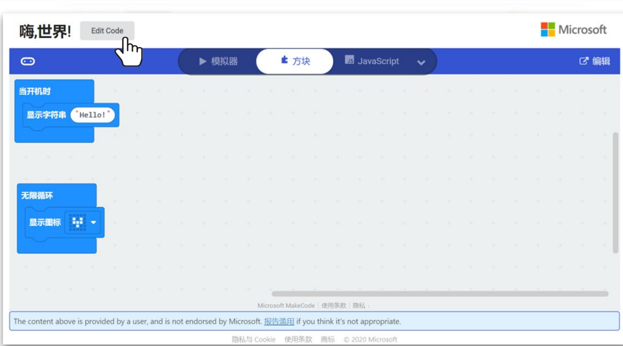
## 快速提示#4！

你可以通过发布项目，然后向他们发送项目网站来与你的老师或朋友共享代码。 为此，请单击［分享］，然后在弹出窗口中单击［发布项目］。 然后，你将看到一个带有项目 URL 的新弹出窗口。 单击［复制］按钮以复制项目网站。



## 快速提示#5！

当你的老师或朋友在浏览器中打开项目网站时，将看到以下页面。 他们可以查看你的代码，也可以通过单击［Edit Code］按钮对其进行编辑。



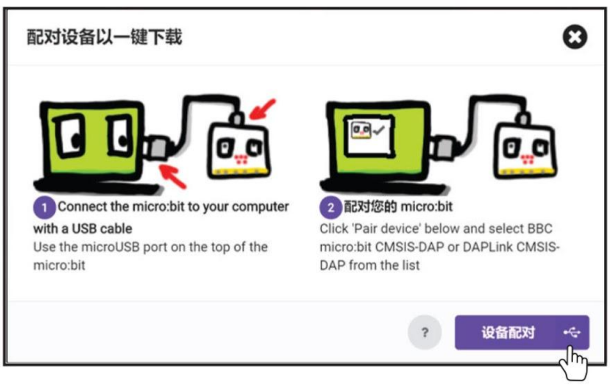
## 快速提示#6！

你知道可以将设备配对以进行一键下载吗？ 为此，请单击齿轮图标，然后选择“设备配对”。


### 注意

你需要在 micro:bit 设备上拥有最新的固件，并使用新的 Edge 或 Chrome 浏览器。 如果你需要更新固件，请按照此处的说明进行操作 – https://microbit.org/

确保你的 EDU:BIT 已连接到 PC，然后在弹出窗口中单击 **[设备配对]** 按钮。 接下来，从列表中选择 BBC micro:bit CMSIS-DAP 或 DAPLink CMSIS-DAP，然后单击 **[Connect]**。





配对设备后，单击 **[下载]** 按钮，可以将代码直接刷新到 EDU:BIT。试试看吧！

如果你在与设备配对时遇到问题，可以参考 https://makecode.microbit.org/device/usb/webusb/troubleshoot 以获得更多信息。

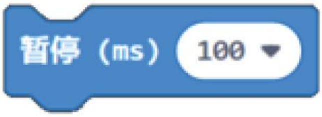
# 探索更多编码块

#1 使用 **[显示 LED]** 块设计自己的图标，并使用 **[显示数字]** 块显示数字。

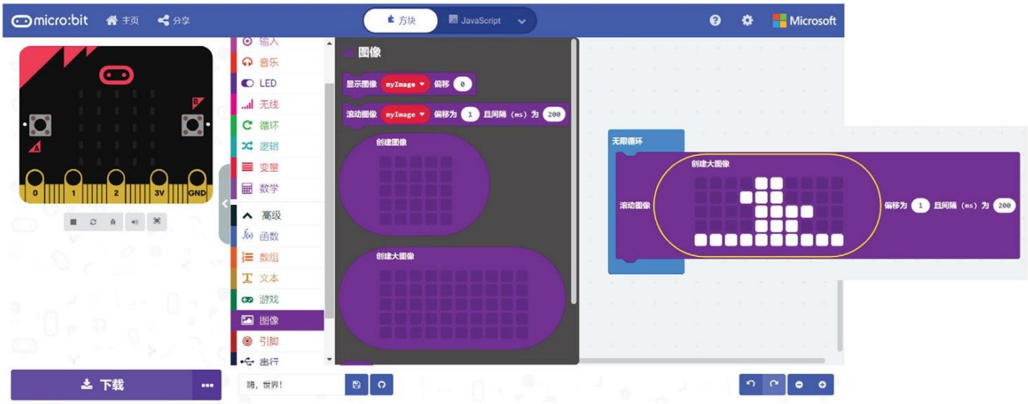


你知道吗？
1000毫秒（ms）= 1秒

#2 添加 **[暂停（ms）__ ]** 块以减慢程序速度。 此功能使程序暂停所设置的毫秒数。



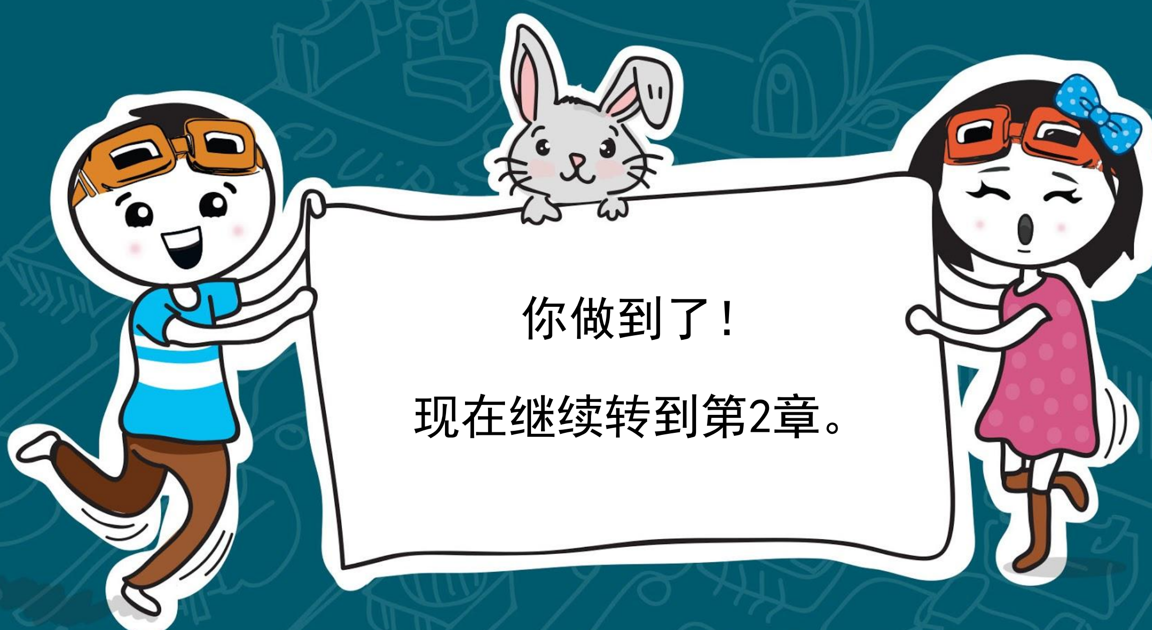

#3 要在LED矩阵显示器上滚动图像，可以使用 **[滚动图像 myImage 偏移为且间隔_（ms）为_ ]** 块，以及 **[图像]** 类别抽屉（在“高级”类别下）的 **[创建图像]** 块或 **[创建大图像]** 块。




运行该程序时，你会看到小鸭子在LED矩阵显示器上一个接一个地移动。

# 编码挑战

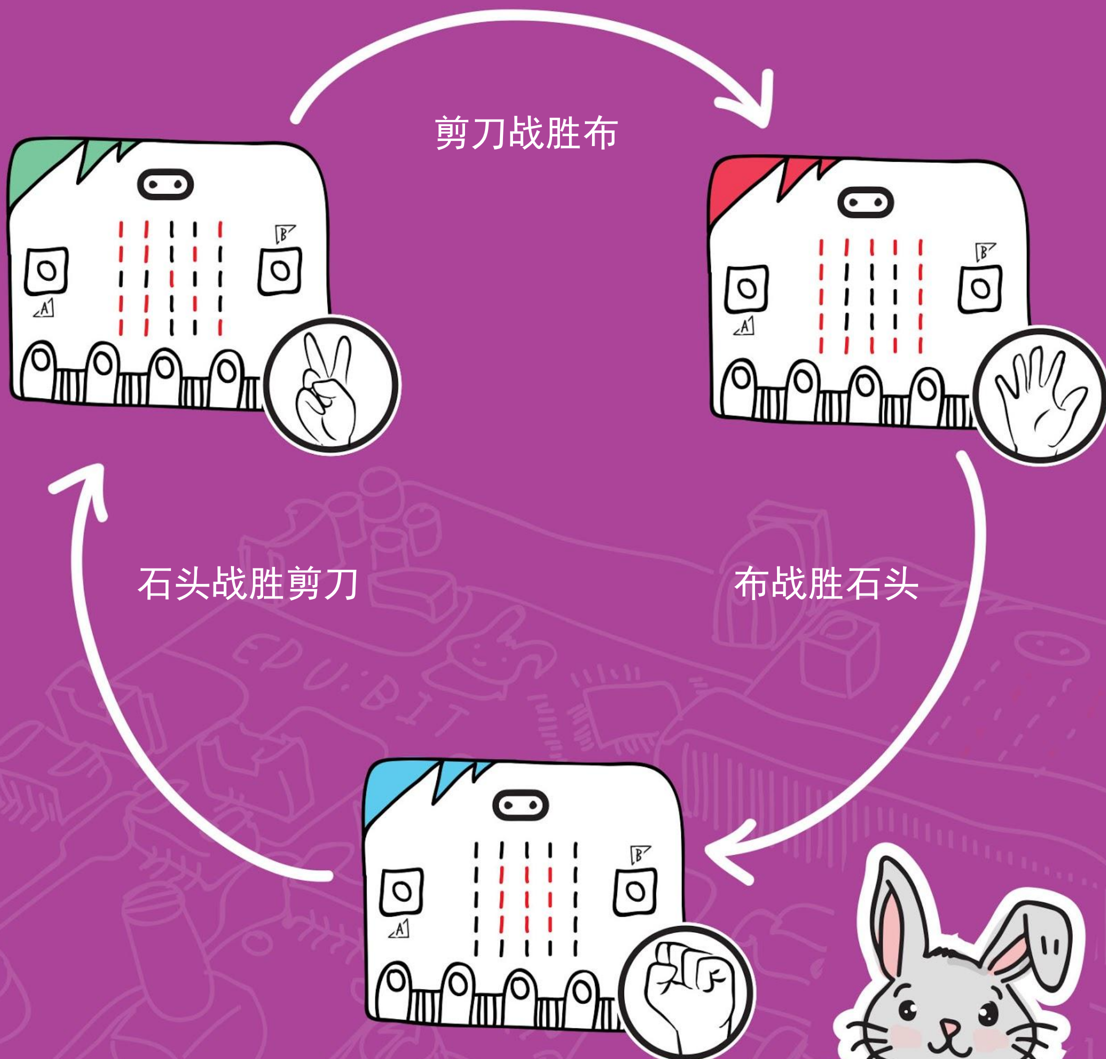
| 将 EDU:BIT 编程为数字公告板。 | |
|---|---|
| 开始时<br>[当开机时] | 显示一个简单的动画以引起注意，然后滚动你的班级名称。 |
| 永远<br>[无限循环] | 滚动今天的日期和该课程的其他重要信息。 |



# 第2章

# 让我们玩剪刀石头布

## Button Bit 和 micro:bit 上的按钮




link.cytron.io/edubit-chapter-2

## 让我们编码！

第1步 转到 https://makecode.microbit.org（或者如果已经在MakeCode编辑器中，则单击［主页］图标）并创建一个新项目。 单击［输入］类别，然后选择［当按钮 __ 被按下时］块上。

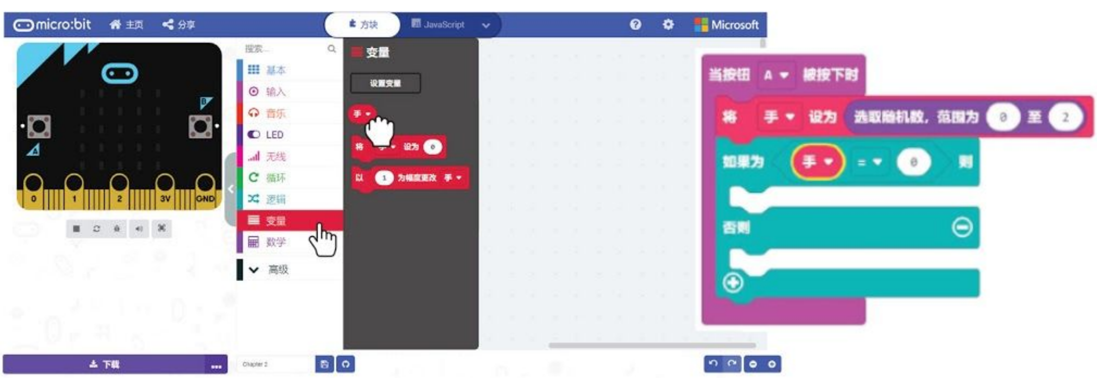

第2步 单击［变量］类别，然后选择［设置变量］。 在弹出窗口中键入“手”，然后单击“确定”。


第3步 单击［变量］类别，然后选择［将 __ 设为 __］块。 将块锁定到［当按钮 A 被按下时］块。

第4步 单击［数学］类别然后选择［选取随机数，范围为 ＿ 至 ＿ ］块。把数目 10 换成 2。


第5步 单击［逻辑］类别然后选择［如果-则-否则］块和［ _=_ ］的对比块。最后把对比块放在“如果”的插槽里。

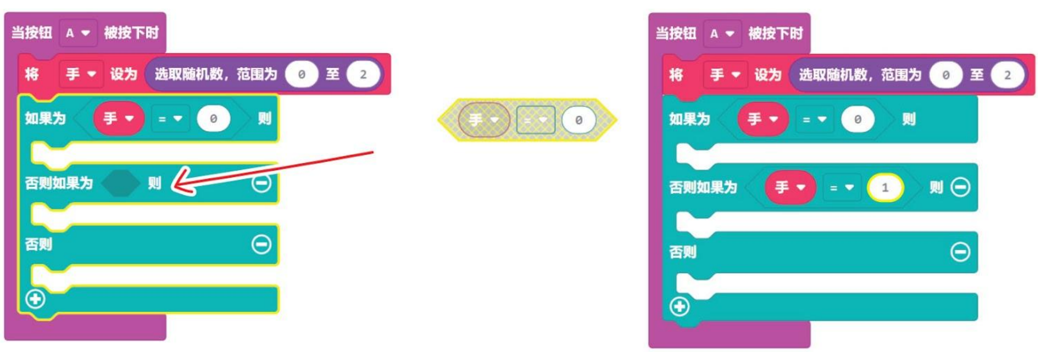

第6步 单击［变量］然后选择［手］块。把它移去放在对比块。



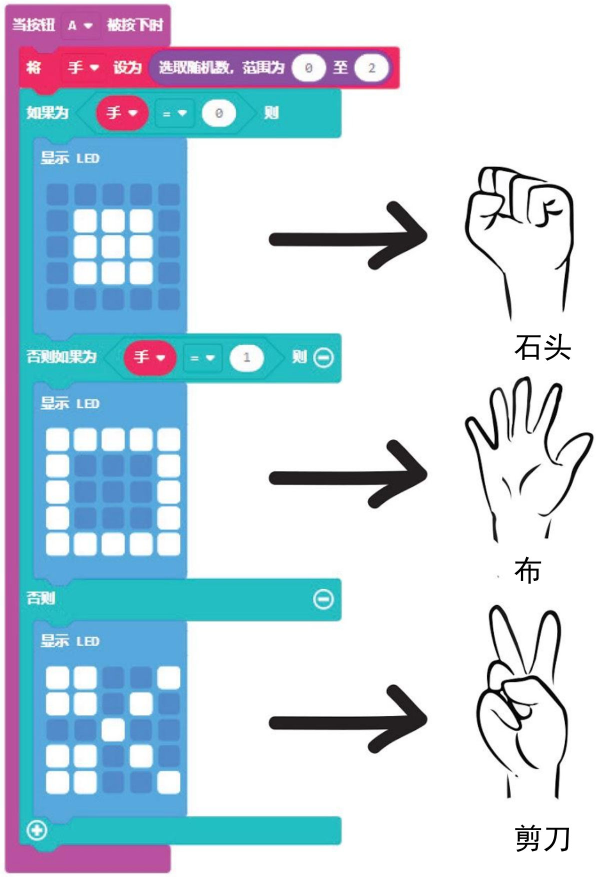
第7步 单击在 + 的标志上加［否则如果］状态在“如果”块。


第8步 右单击在对比块然后选择“**重复**”。


第9步 把复制块移去“否则如果”插槽然后把数目 0 换成 **1**。



第10步 加［基本］:［显示 LED］块去 “如果”，“否则如果”和“否则”插槽。单击箱盒在［显示 LED］块然后根据以下图表例子制作图案。

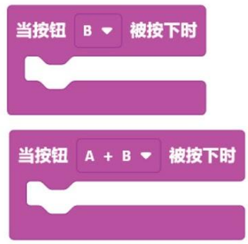




闪烁你的编码在 EDU:BIT 然后你就可以和朋友玩剪刀石头布。每当你按下按钮 A 在 micro:bit 或黄色按钮，LED 矩阵将随机显示：“石头”，“布” 或“剪刀”。

**温馨提醒：**
如果你要收藏这项目，记得按储存按钮然后保存在你的电脑文件夹里。

让我们为每个玩家分配 3 条生命来改进此游戏。为此，你需要创建一个名为“生命”的新变量并添加以下代码块。

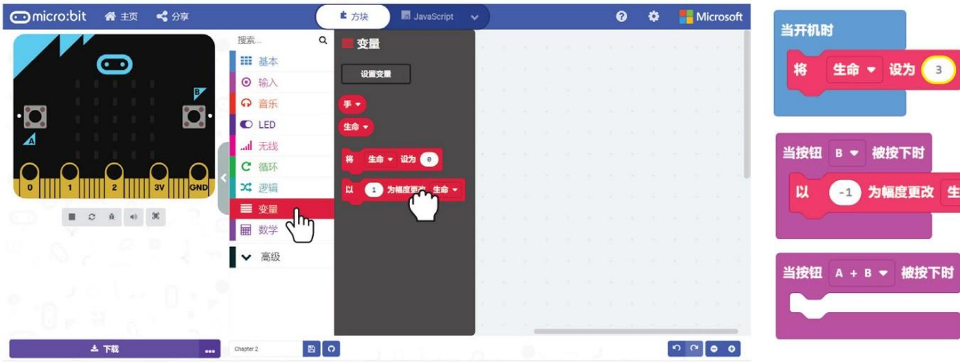
第11步 单击［输入］类别，然后选择［当按钮 __ 被按下时］块上。 复制该块并将设置分别更改为“B”和“A+B”。



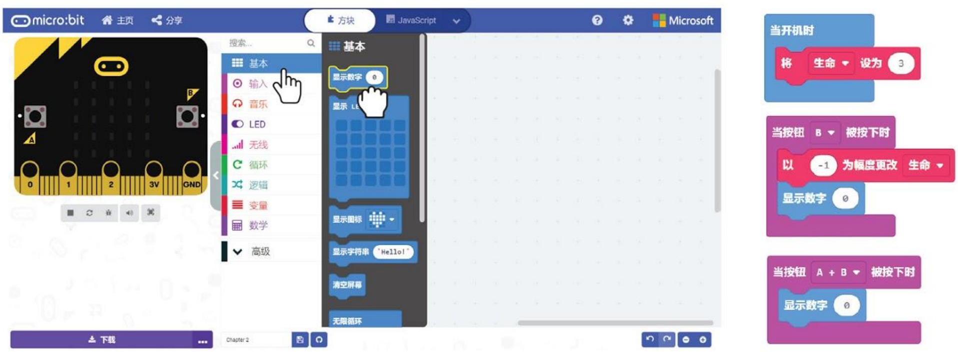

第12步 单击［变量］类别，然后选择［设置变量］。 在弹出窗口中“生命”，然后单击“确定”。


第13步 单击［变量］类别，然后选择［将 __ 设为 __］块。 将块对齐到［基本］:［当开机时］块。 将变量设置为“生命”，并将其值更改为 3。

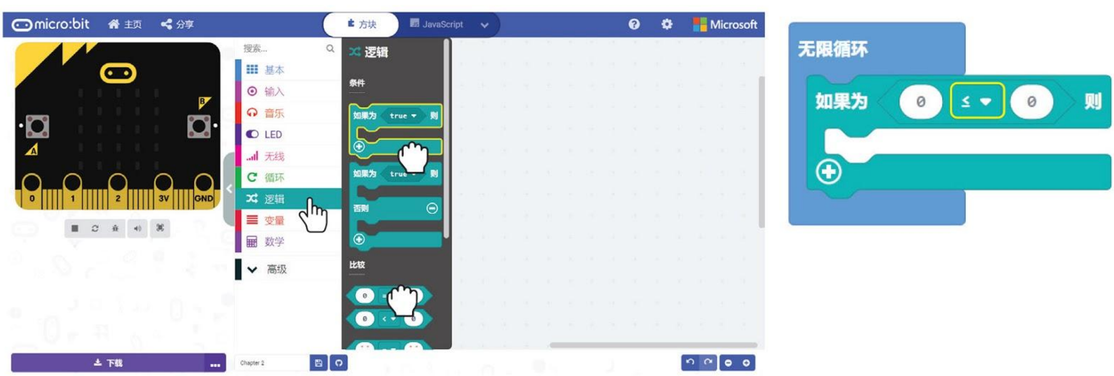
第14步 再次单击［变量］类别，然后选择［以 __ 为幅度更改 __ ］块。 将块锁定到［当按钮 B 被按下时］块。 将变量设置为“生命”，并将其值更改为 -1。



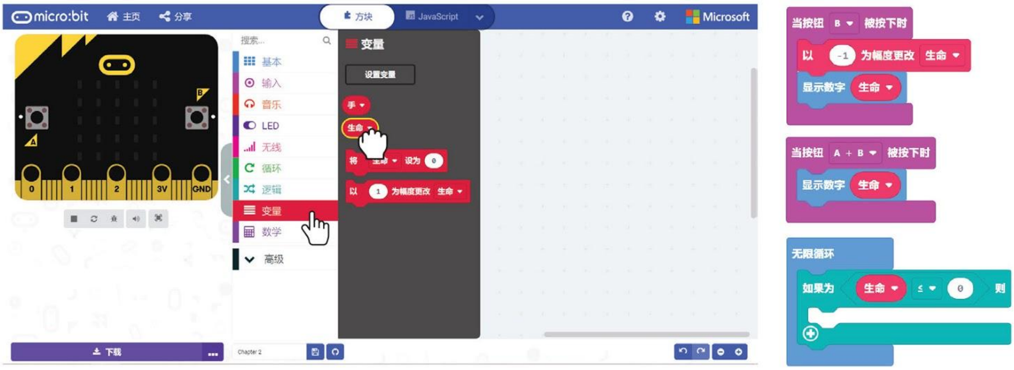

第15步 单击［基本］类别，然后选择［显示数字］块。 复制它，并将此块吸附到两个［当按钮 __ 被按下A+B时］块上。




第16步 单击［逻辑］类别，然后将［如果-则］块和［ _=_ ］比较块添加到你的代码中。 将它们捕捉到［基本］:［无限循环］块，并将符号更改为“≤”。

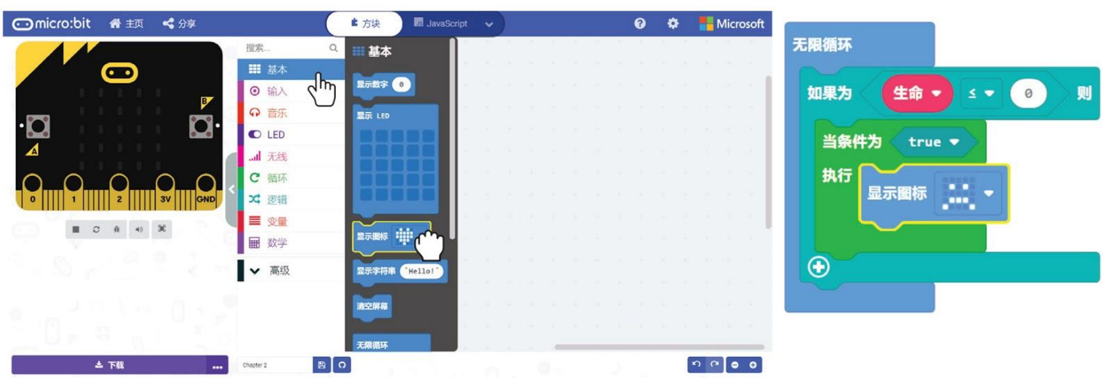


第17步 单击**［变量］**类别，然后选择**［生命］**块。 复制并捕捉到**［显示数字］**块以及**［ __ = __ ］**比较块的左侧插槽。

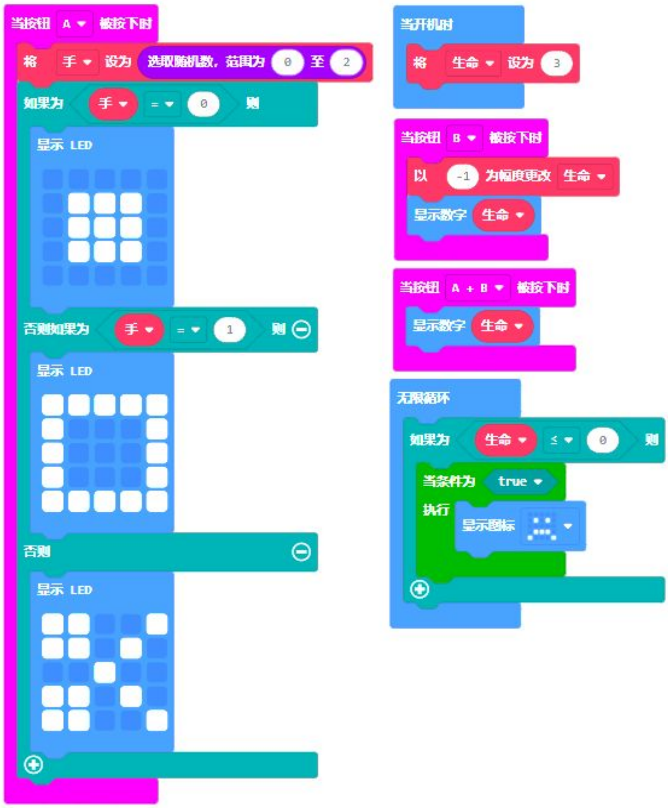



第18步 单击**［循环］**类别，然后选择**［当条件为 __ 执行］**块。 将其卡入**［如果-则］**块的插槽。

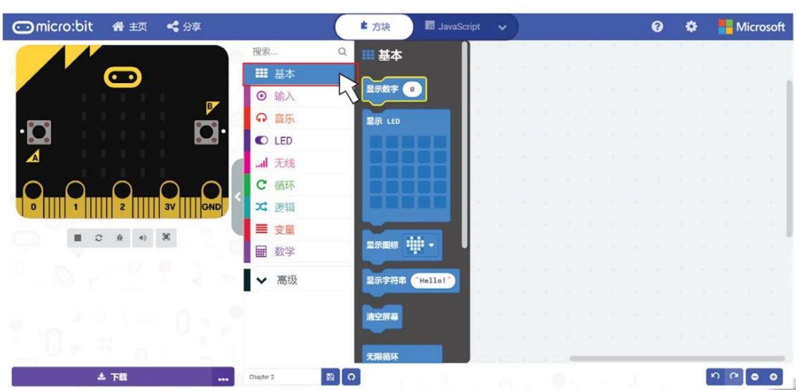

第19步 单击**［基本］**类别，然后选择**［显示图标］**块。 将其捕捉到**［当条件为 __ 执行］**块的插槽中，并将图标更改为“悲伤的面孔”。



第20步 这是完整的代码。 将其刷新到你的 EDU:BIT，并与你的朋友一起玩，以确定剪刀石头布游戏的获胜者。

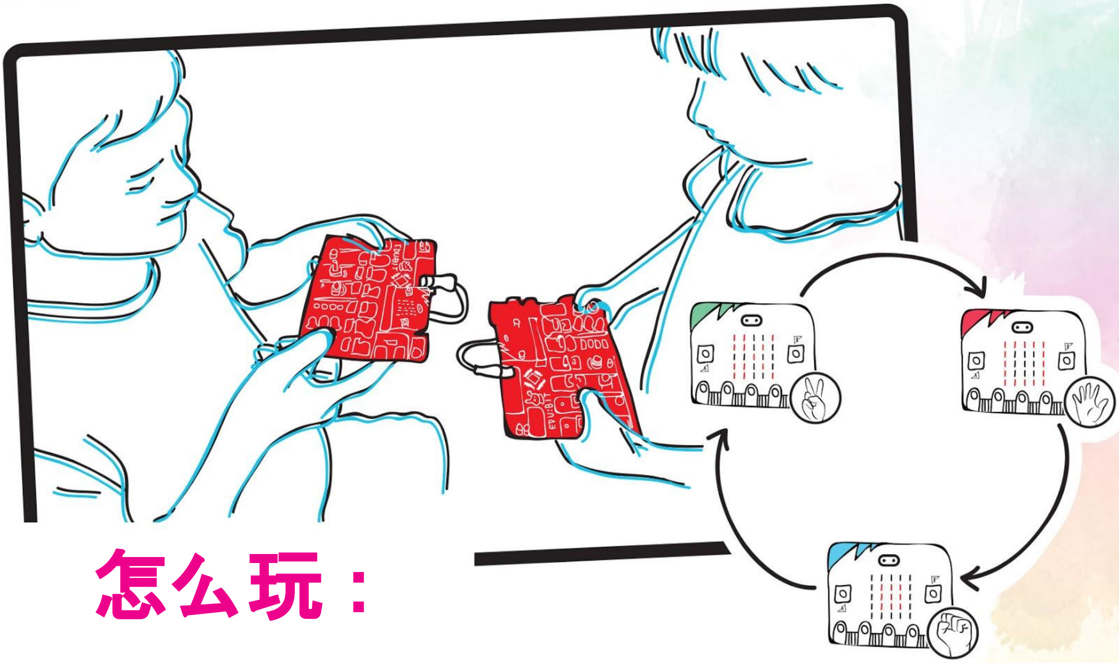






# 让我们一起玩

## 剪刀石头布高级版



## 怎么玩：

站起来面对对手。当两个玩家都准备就绪时，按黄色按钮（按钮A）随机显示剪刀，石头，布。

比较并决定谁获胜。

如果输了，则需要按一次 EDU:BIT上的蓝色按钮（按钮B）以减少一次生命。

同时按下黄色和蓝色按钮以检查并显示剩余寿命。

如果玩家输了3次，则游戏结束，他/她的 EDU:BIT 将显示悲伤的表情。

**注意：**

- 要进行另一回合，你需要 [RESET] 重置你的板子以重新开始。
- 如果没有朋友可以玩，则可以随时在 MakeCode 编辑器上与模拟器进行游戏。

# 破解密码

在计算机编程中，我们使用 **变量** 来存储可以在运行时（程序运行时）更改的信息或值。 你可以将变量视为带标签的信封，其中包含一张上面写有信息的纸。 可以取出该纸并将其替换为具有新信息的另一张纸。

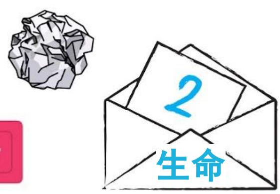
在前面的代码中，我们创建了一个名为“生命”的变量，并为其分配了初始值3。



写在纸上的数字3是信息

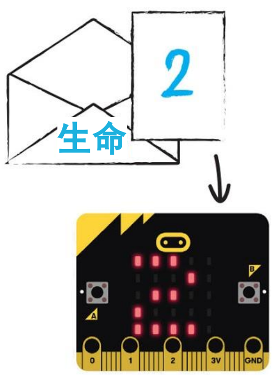
上面写有 “生命” 的信封是变量。

然后，每当按下按钮B时，该值就会被改变-1。


按下按钮B时，从信封上取下写有 “3” 的纸，并换成写有 “2” 的另一张纸（3 – 1 = 2）。

同时按下按钮 A + B 将导致 LED 矩阵显示变量“生命”的当前值。

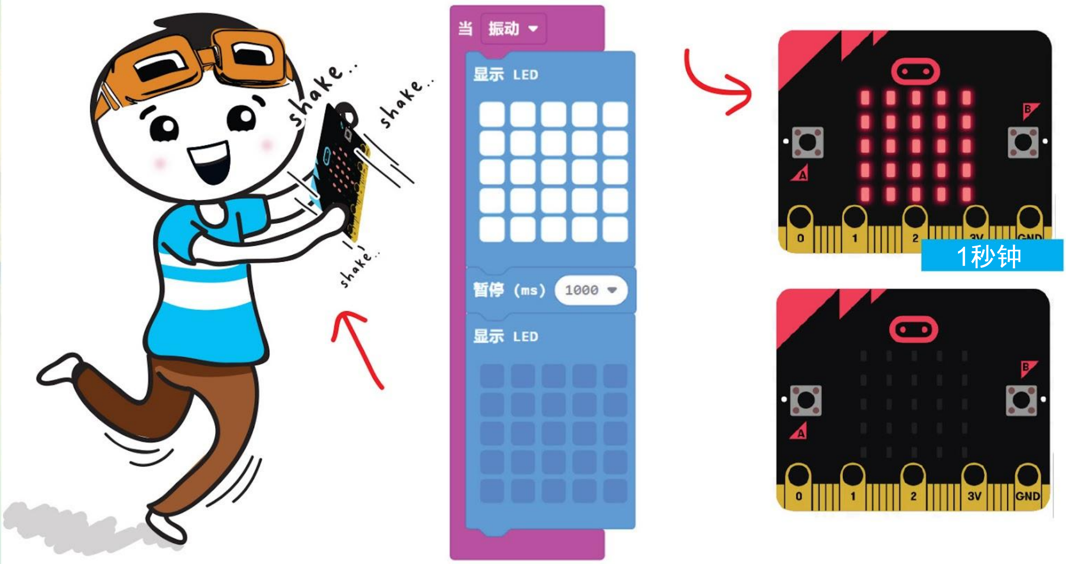




当按下按钮 A + B 时，纸张被从信封中拉出，并且上面写的信息被“读取”

# 探索更多编码块

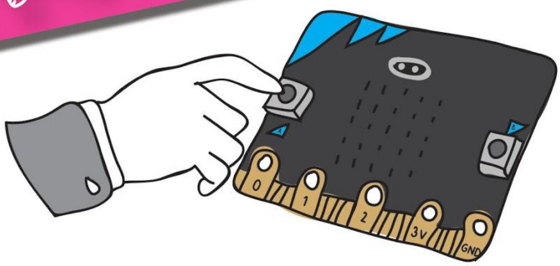
除了 **[当按钮 __ 被按下时]** 块之外，你还可以将 **[输入]** 类别抽屉中的其他块用于基于事件的编程。用户执行的操作（例如按下按钮或摇动面板）被称为 “事件”。

例如，以下代码将在每次摇动板时导致LED矩阵点亮1秒钟。我们试试吧~

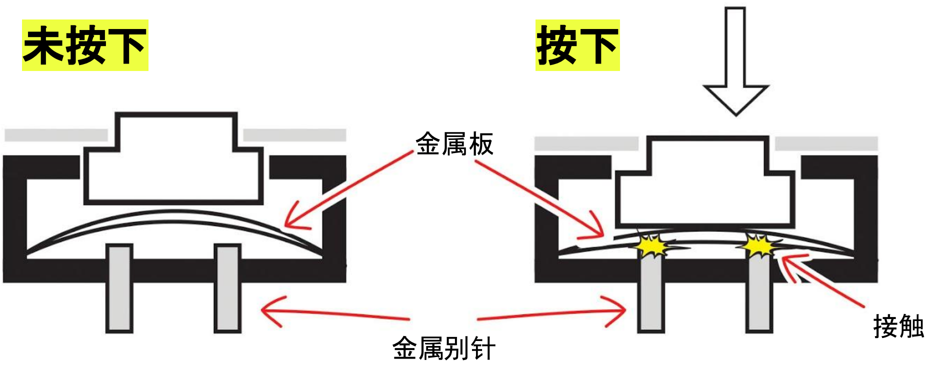


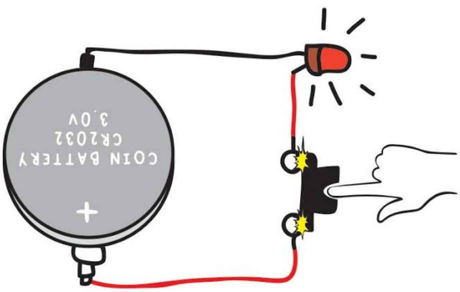

如果单击块上的 **[振动]** 按钮，将出现一个弹出画面，显示其他触发器的一部分。尝试对 EDU:BIT 进行编程，以针对每种情况显示不同的图标。探索愉快！


# 有趣的事实！




**按钮**是输入设备或开关，只有两种可能的状态 -- 按下或未按下。

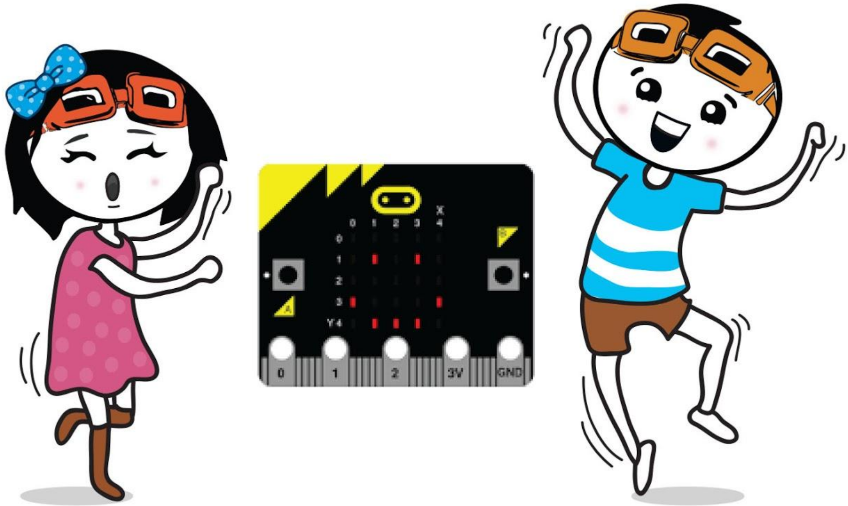



按下按钮后，电路完成，LED点亮！ 猜猜当你松开按钮时会发生什么？


黑色，灰色，绿色和白色按钮通常用于开/关功能，红色用于紧急按钮或机械紧急停止。

# 编码挑战




<table>
<tr><td colspan="2">程序 EDU:BIT 用作记录学生出勤率的计数器。 女孩进入班级时必须按按钮 A ； 和男孩进入班级时，必须按按钮 B。</td></tr>
<tr><td>当开机时</td><td>露出笑脸。<br>设置变量 Girl=0 和 Boy=0。</td></tr>
<tr><td>当按钮 A 被按下时<br>（黄色按钮）</td><td>将变量 Girl 增加 1</td></tr>
<tr><td>当按钮 B 被按下时<br>（蓝色按钮）</td><td>将变量 Boy 增加 1</td></tr>
<tr><td>当按钮 A+B 被按下时<br>（黄色和蓝色按钮）</td><td>在LED显示屏上滚动以下信息：<br>Total=(Girl+Boy) ; Girl=(Girl) ; Boy=(Boy)</td></tr>
</table>

# 第3章

## 来点音乐吧~

Music Bit（压电式蜂鸣器，音频插孔）

## 让我们编码！

第1步 在你的 MakeCode 编辑器中创建一个新项目。单击［输入］类别，然后选择［当按钮 __ 被按下时］块上。



第2步 单击［音乐］类别，然后选择［播放旋律 __ 重复 __ ］块。



第3步 单击［dadadum］，然后从下拉列表中选择**“生日歌”**旋律。



单击屏幕模拟器上的 Button A。 你听到熟悉的曲调了吗？ 也乐于尝试其他旋律~
*确保你的计算机扬声器已打开。

除了预设旋律列表外，你还可以对EDU:BIT进行编程以播放任何喜欢的歌曲。 让我们尝试播放这种动听的音乐。

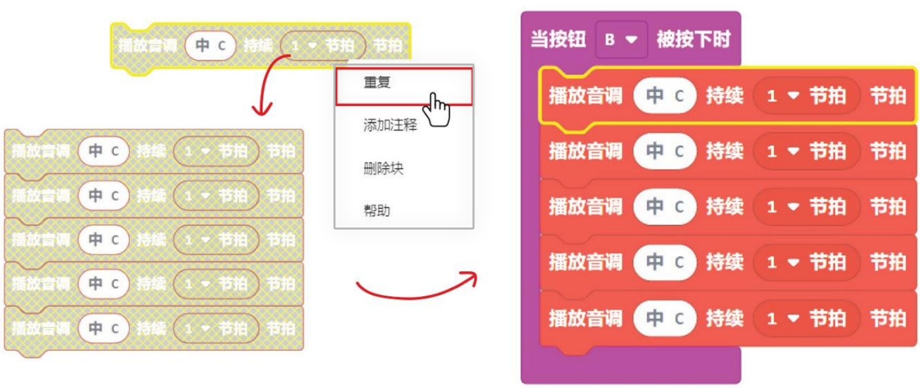
第4步 单击［输入］类别，然后选择［当按钮 __ 被按下时］块上。 选择按钮“B”。



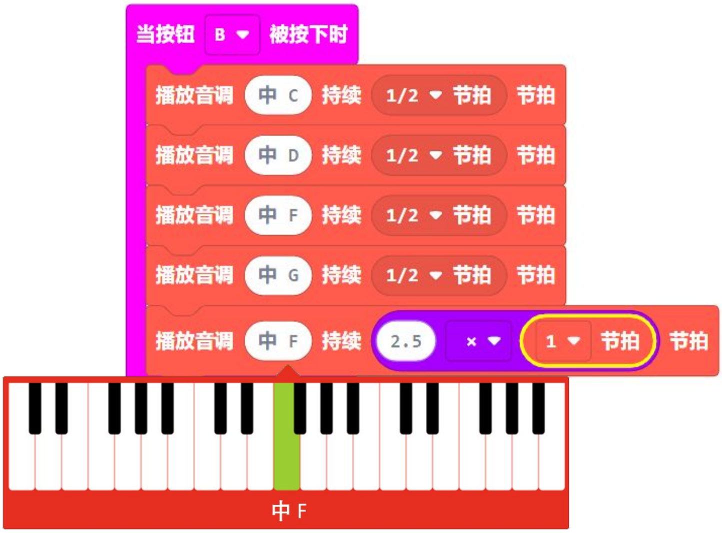
第5步 单击［音乐］类别，然后单击［播放音调 __ 據 __ 节拍 ］块。



第6步 在工作场所中，右键单击［播放音调 __ 據 __ 节拍 ］块，然后单击“重复”。 重复直到共有五个［播放音调 __ 據 __ 节拍 ］块。将块放在［当按钮 A 被按下时］块的插槽中。

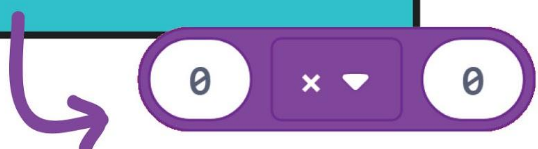


第7步 根据以下示例代码，选择这些 **[播放音调 __ 据 __ 节拍 ]** 块的 '**音调**' 和 '**节拍**'：




## 注意！

紫色块取自 [**数学**] 类别抽屉。




单击屏幕模拟器上的按钮B。 你能猜出这首歌吗？

进行编程时，建议你定期检查代码，以确保你处在正确的轨道上。 你可以将模拟器用于此目的。

第8步 通过添加更多的 **[播放音调 __ 据 __ 节拍 ]** 块并相应地更改 '**音调**' 和 '**节拍**'，继续编码歌曲的其余部分。 你可以参考下一页的音调和节拍。


## 我会跟随你

# I Will Follow You

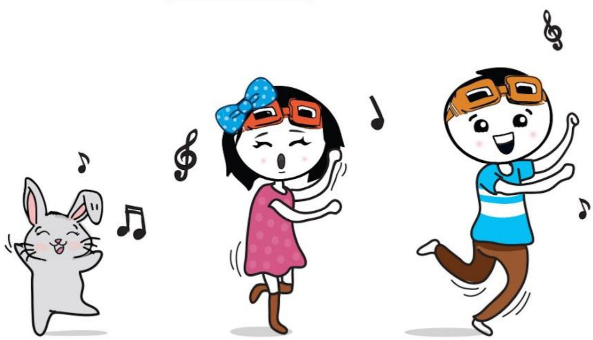

I
will
fol-
low
you,

Fol-
low
you
wher-
ev-
er
you
may
go,

There
is
n't
an
o-
cean
too
deep

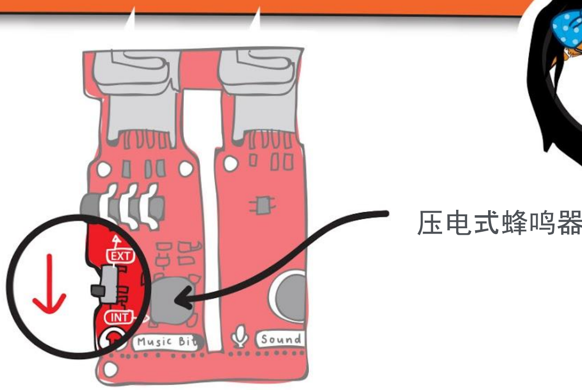
A
moun-
tain
so
high
it
can
keep,

Keep
me
a-
way
…
…

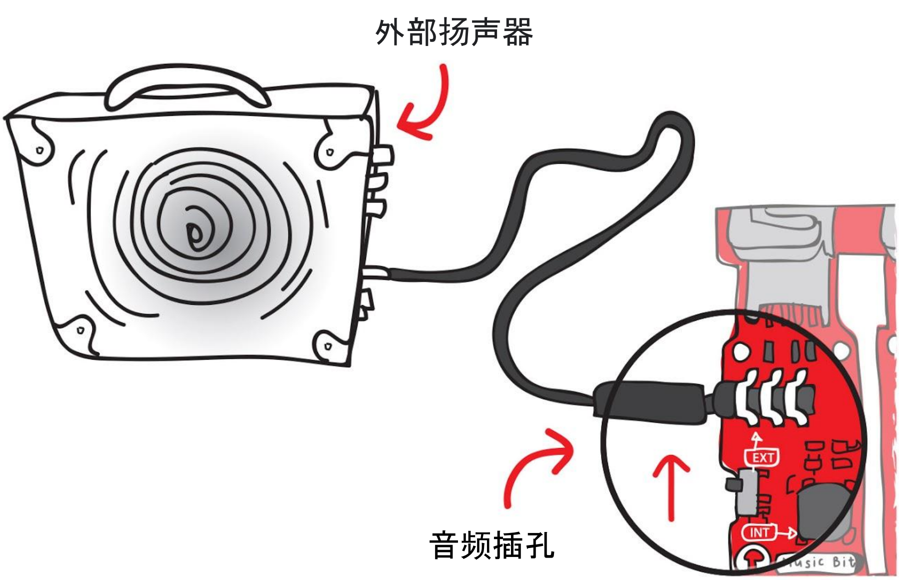



第9步 将完成的代码刷新到你的 EDU:BIT 。

每当你按下 EDU:BIT上的蓝色按钮（按钮B）时，EDU:BIT就会播放歌曲“我会跟随你”。 切记打开 EDU:BIT 的电源并通过将开关滑动到 INT（内部）来打开压电式蜂鸣器。

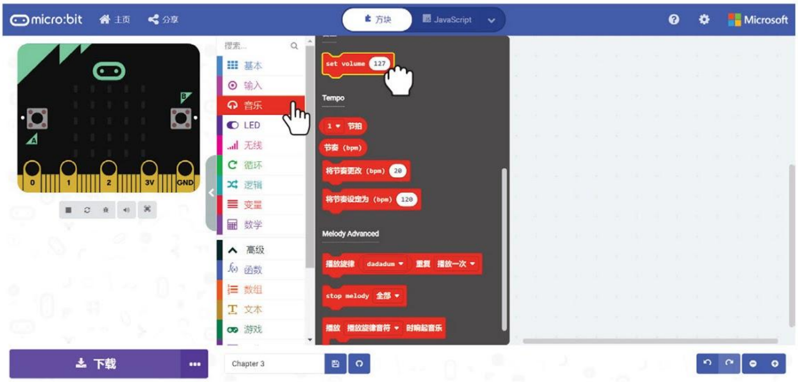



或者，你可以将外部扬声器或耳机连接到 EDU:BIT上的音频插孔。你需要将开关滑动到 EXT（外部）。



音乐太柔和了吗？还是太大声？你可以添加 **[设置音量 __ ]** 块来更改音量，范围从 0 到 255 (最大响度)。

第10步 单击 **[音乐]** 类别，然后选择 **[设置音量 __ ]** 块。将块对齐到 **[当开机时]** 块并将音量更改为 **200**。



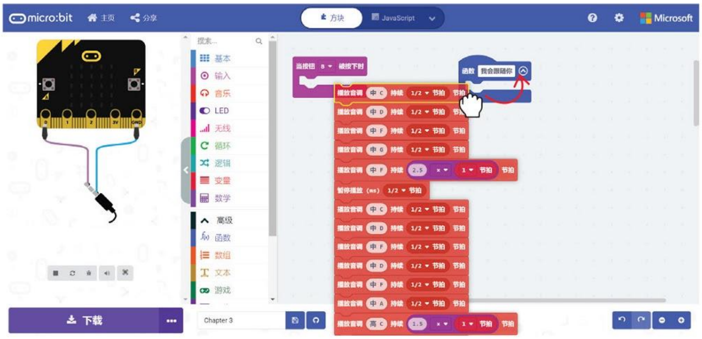


## 注意！

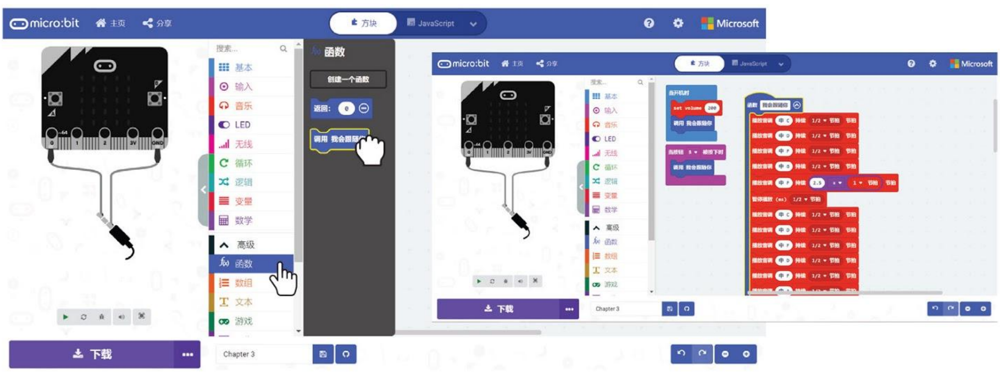
我们可以将执行特定任务的代码块变成一个**函数**，例如，你的代码播放“我会跟随你”歌曲。 在编程中，功能是指例程或一组过程。 定义函数后，就可以在程序中的多个位置使用它，而不必一遍又一遍地重写相同的代码块。

第11步 单击 **[高级]** 类别，然后选择 **[函数]** 类别。单击 **[创建一个函数]**，然后在弹出窗口中将 doSomething 重命名为 “**我会跟随你**”。然后点击“完成”。

第12步 [调用我会跟随你] 块将出现在编辑器上。 单击 [当按钮 B 被按下时] 块中最上面的块，按住并将所有块拖到 [调用我会跟随你] 插槽中。



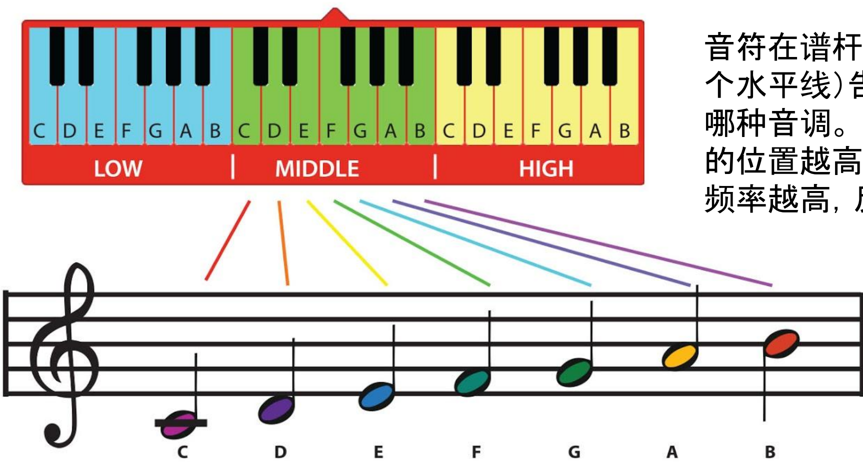

第13步 单击 [函数] 类别，然后选择 [调用我会跟随你] 块。 复制块。 将 [调用我会跟随你] 块对齐到 [当开机时] 块和 [当按钮 B 被按下时] 块。这是示例代码：




第14步　将完成的代码刷新到你的 EDU:BIT。 请欣赏音乐~

# 破解密码

如果你知道如何阅读音乐，则可以将 EDU:BIT 编程为播放其他歌曲。 这是一个简单的指南，可帮助你“解码”乐谱。




音符在谱杆上的位置(即五个水平线)告诉我们要播放哪种音调。音符在五线谱上的位置越高，声音的音高或频率越高，反之亦然。

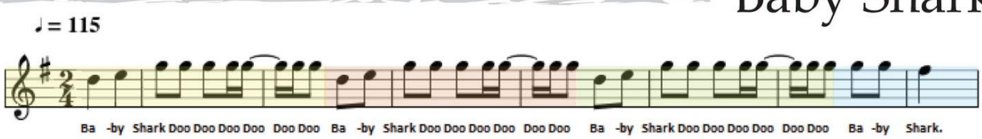
| Sign | Rest | Relative Length | Duration |
|---|---|---|---|
| 𝅝 | 𝄻 | Whole Note | 4 beats |
| 𝅗𝅥 | 𝄼 | Half Note | 2 beats |
| ♩ | 𝄾 or 𝄽 | Quarter Note | 1 beat |
| ♪ | 𝄾 | Eighth Note | 1/2 beat |
| 𝅘𝅥𝅯 | 𝄿 | Sixteenth Note | 1/4 beat |

使用不同的乐谱来告诉我们音符的演奏时间(即多长时间)。

# 破解密码

使用给出的线索，你可以“解码”以下曲调吗？

## Baby Shark



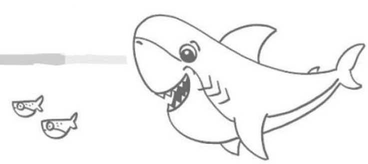

| Line 1 | Ba | -by | Shark | doo | doo, | doo | doo | doo | doo |
|---|---|---|---|---|---|---|---|---|---|
| Note | High D | High E | High G | | High G | High G | High G | | High G |
| Beat | 1 | | 1/2 | 1/2 | 1/2 | | 1/2 | 1/4 | 1/2 |

| Line 2 | Ba | -by | Shark | doo | doo, | doo | doo | doo | doo |
|---|---|---|---|---|---|---|---|---|---|
| Note | High D | | High G | High G | High G | | High G | High G | High G |
| Beat | 1/2 | 1/2 | 1/2 | | 1/2 | 1/4 | 1/2 | | 1/2 |

Repeat the same for Line 3

| Line 4 | Ba | -by | Shark |
|---|---|---|---|
| Note | | High G | High F# |
| Beat | 1/2 | 1/2 | |

当同时按下黄色按钮（按钮A）和蓝色按钮（按钮B）时，EDU:BIT 将播放Baby Shark 曲调。



**注意!**

使用 **[设置音量 __ ]** 块可以调节音乐的响度。

# 探索更多编码块

**#1** 你可以使用 **[将节奏设定为(bpm) __ ]** 块来设置 “速度”（即歌曲的速度）。每分钟的节拍 (bpm) 越高，你的音乐越快或更活泼。使用 **[将节奏更改（bpm) __ ]** 块来更改速度。

**#2** 使用 **[停止旋律 __ ]** 块可停止当前正在播放的旋律。

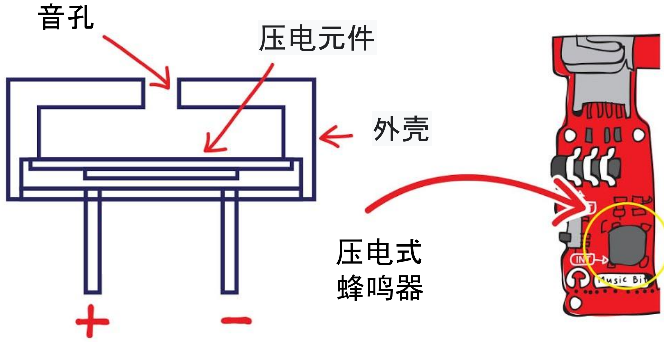
**#3** 你也可以使用 **[播放 __ 时响起音乐]** 及其条件选择，例如开始进行旋律和结束旋律，作为代码中的事件触发。

**这是一个示例代码**：


1. 在此程序中，初始速度设置为120 bpm。
2. 旋律开始时，LED 矩阵将显示音符图标。
3. 旋律结束时，矩阵显示屏上的所有 LED 都将关闭。
4. 每次按下按钮 A，速度都会提高 50 bpm。
5. 每当按下按钮 B 时，都会播放旋律 “entertainer”。
6. 同时按下按钮 A + B 时，旋律停止。

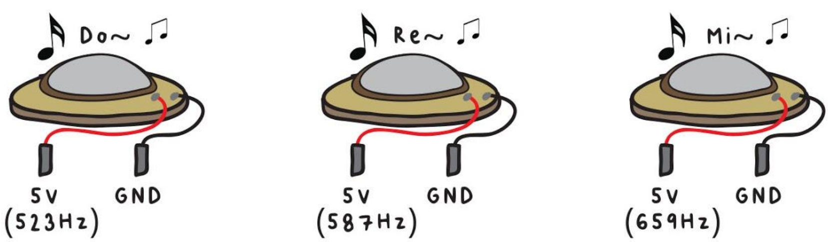
# 有趣的事实！

**压电式蜂鸣器 (Piezo Buzzer)** 通常用于通过电信号通过压电元件时产生振动来产生声音。




通过改变电信号的频率，振动的速度就改变了。因此，压电式蜂鸣器 (piezo buzzer) 会产生不同音调的声音。

**压电元件**

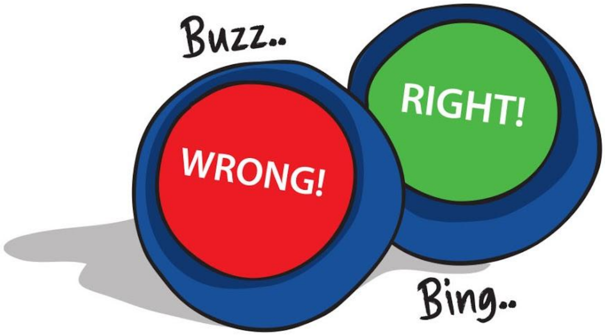



人耳可以听到 20Hz 至 20,000Hz 频率范围内的声音。任何低于 20Hz 的声音称为次声，而高于 20,000Hz 的声音称为超声。

进一步学习！

youtu.be/cxfPNc4Wefo

# 编码挑战




| | |
|---|---|
| 编程 EDU:BIT 用作游戏节目蜂鸣器，以发出 正确/错 误答案的信号。 | |
| **当开机时** | 显示笑脸 |
| **当按钮 A 被按下时**（黄色按钮） | 显示 “✅” 图标并播放一次 “power up” 旋律。 |
| **当按钮 B 被按下时**（蓝色按钮） | 显示 “❌“ 图标并播放一次 “wawawawaa” 旋律。 |
| **当按钮 A+B 被按下时**（黄色和蓝色按钮） | 刷新屏幕 |

# 第4章

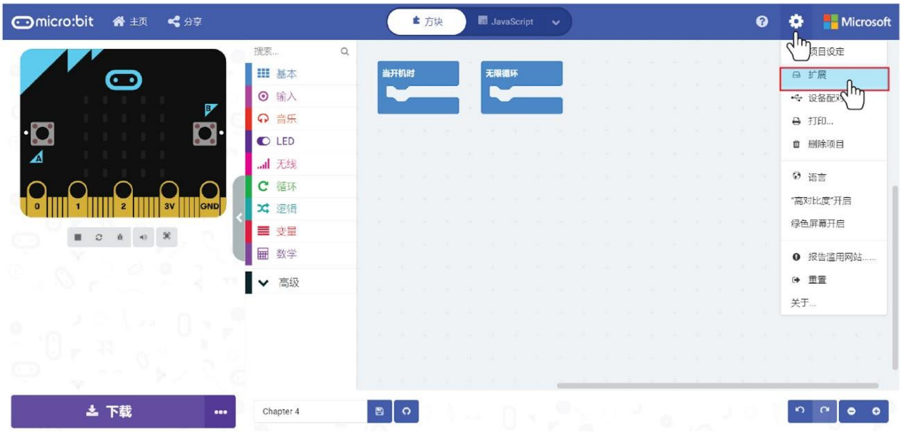
# 赢，输或和

(Traffic Light Bit)

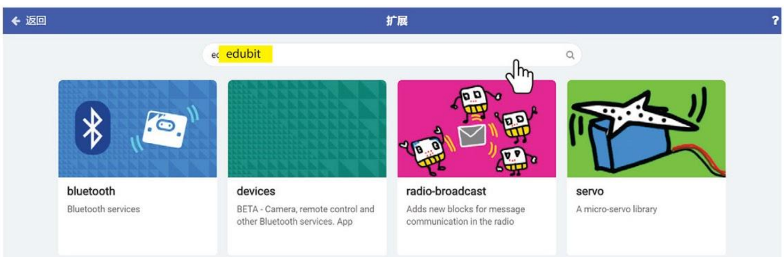
你是否在EDU:BIT上看到一组红色，黄色和绿色的LED？ 那就是 Traffic Light Bit 。要对其进行编程，你需要在MakeCode 编辑器中添加 EDU:BIT 扩展名。扩展程序是我们添加到编辑器中的自定义模块集，使我们能够轻松地编程 micro:bit 附件，例如我们的 EDU:BIT板。

## 让我们编码！

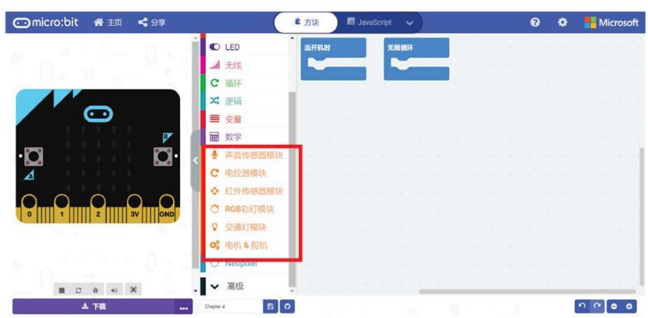
第1步 在你的MakeCode编辑器中创建一个新项目。单击齿轮图标，然后选择 “**扩展**”。*你需要 Internet 连接才能添加扩展名。




第2步 在搜索框中输入“**edubit**”，然后单击 Enter。



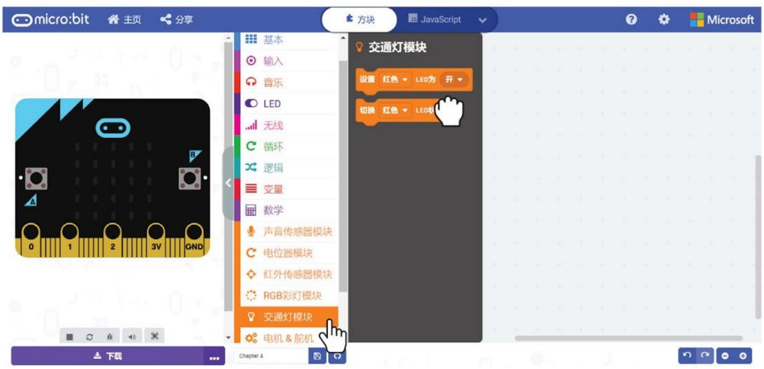

第3步 点击 “edubit” 扩展名。等待它加载，你将在 MakeCode 编辑器中注意到以下新类别的抽屉。

第4步 单击 **[输入]** 类别，然后选择 **[当按钮 __ 被按下时]** 块上。

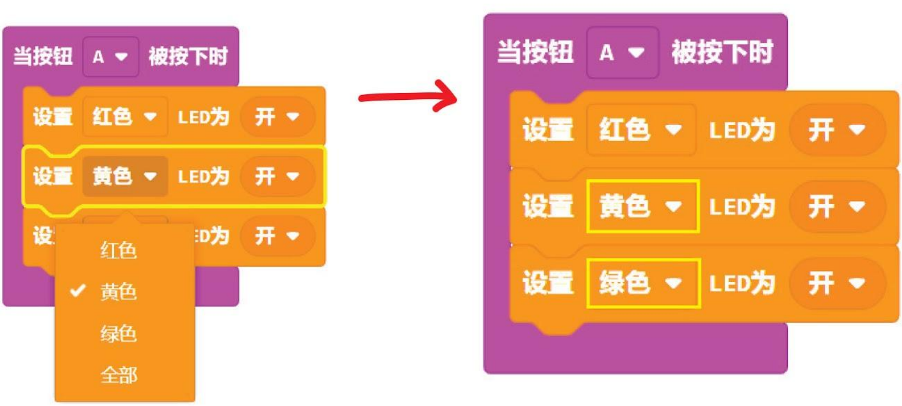


第5步 单击 **[交通灯模块]** 类别，然后选择 **[设置 __ LED为 __]** 块。在工作场所中，右键单击 **[设置 __ LED为 __]** 块，然后单击“重复”。重复直到有三个 **[设置 __ LED为 __]** 块。将块吸附到 **[当按钮 A 被按下时]** 插槽。

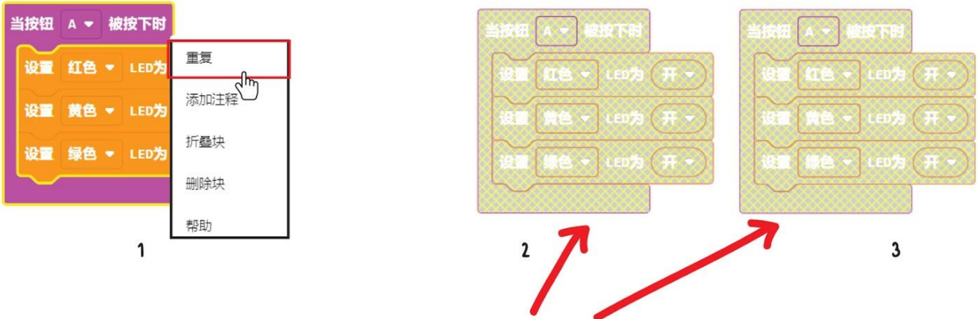




第6步 单击颜色选择，然后将第二和第三块分别更改为“**黄色**”和“**绿色**”。



第7步 右键单击 [当按钮 A 被按下时] 块，然后选择 “**重复**”。重复以获取三组相同的块。

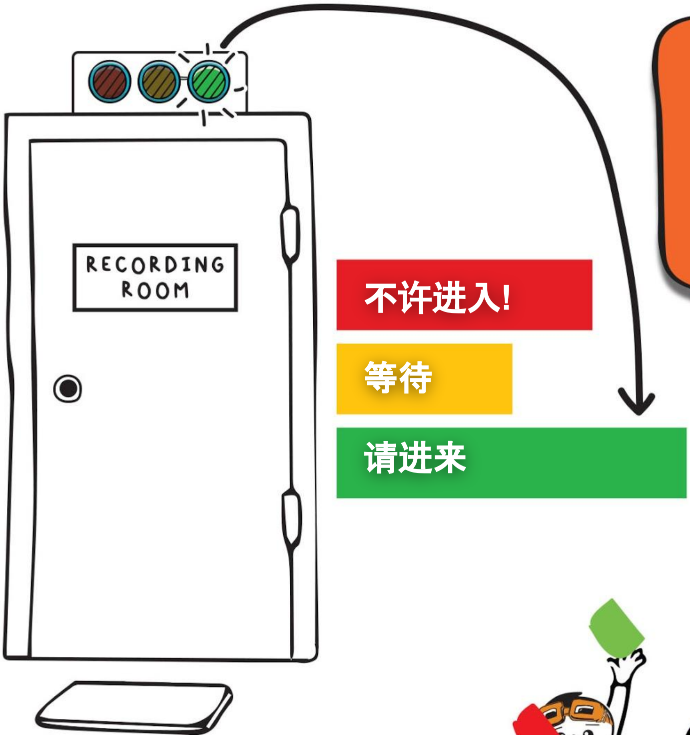


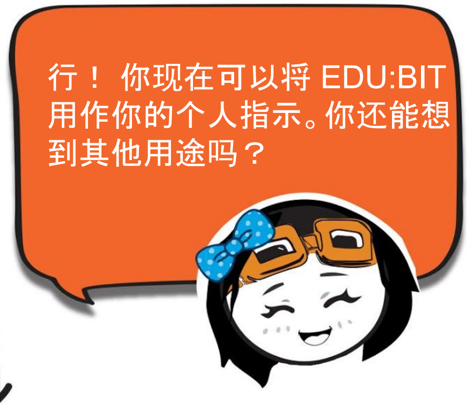

*由于有多个[当按钮 A 被按下时]块，因此这些块被禁用并且不会运行。

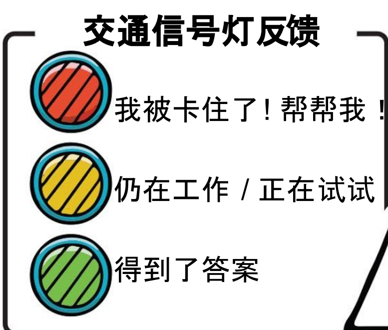
第8步 将第二个和第三个 [当按钮 A 被按下时] 块上的 “A” 分别更改为 “**B**” 和 “**A + B**”。

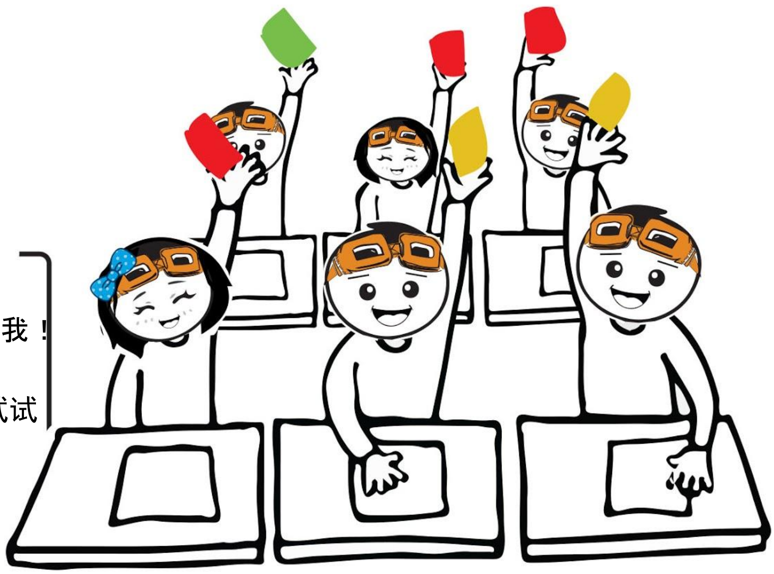
第9步 如下将 LED 的状态从打开更改为关闭。

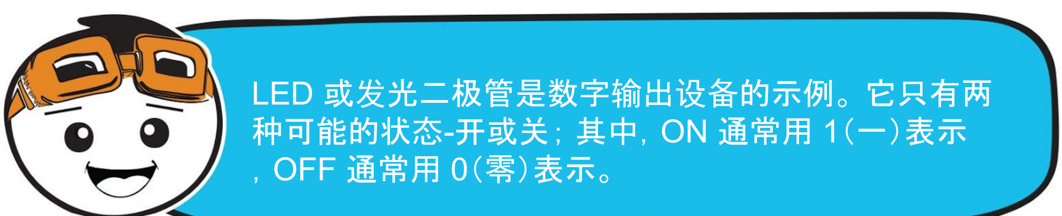

第10步 将代码刷新到你的 EDU:BIT，观察当你同时按下按钮A，按钮B 和两个按钮 A + B 时会发生什么。


RECORDING ROOM
不许进入!
等待
请进来


行！ 你现在可以将 EDU:BIT 用作你的个人指示。你还能想到其他用途吗？


交通信号灯反馈
我被卡住了！帮帮我！
仍在工作 / 正在试试
得到了答案



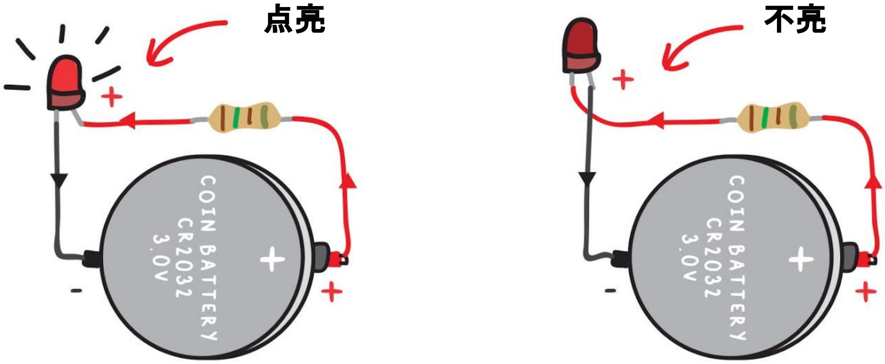

LED 或发光二极管是数字输出设备的示例。它只有两种可能的状态-开或关；其中，ON 通常用 1（一）表示，OFF 通常用 0（零）表示。

你还可以将 EDU:BIT 编程为计时器指示器。
这是示例代码。

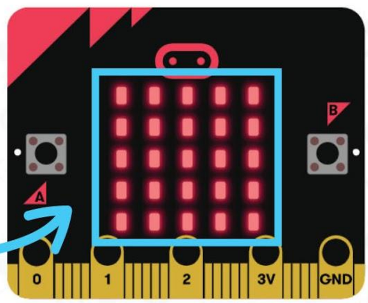

“切换”是指从一种状态切换到另一种状态。如果当前状态为 ON，则它将切换为 OFF，反之亦然。因此，当我们反复切换 LED 时，LED 似乎会闪烁。

# 有趣的事实！

**发光二极管（LED）**是一种半导体器件，可以从电中发光。它具有2个端子，即正极端子和负极端子。当以正确的极性连接 LED 并有电流流过时，LED 会发光。




Micro:bit 中使用的 LED 基于表面贴装技术（SMT），并且可能非常小。




除了 micro:bit 上的这些，EDU:BIT 板上还有另外 41 个 SMT LED。你能全部看到吗？

进一步学习！

youtu.be/qqBmvHD5bCw

# 编码挑战

| | |
|---|---|
| 程序 EDU:BIT 用作得分计数器以及 “赢，输或平/画” 和 “Charade” 等游戏的计时器 | |
| **当开机时** | 设置变量 Team A = 0<br>设置变量 Team B = 0 |
| **当按钮 A 被按下时**<br>（黄色按钮） | 将 Team A 更改为1。<br>显示 Team A 的当前得分。 |
| **当按钮 B 被按下时**<br>（蓝色按钮） | 将 Team B 更改为1。<br>显示 Team B 的当前得分。 |
| **当按钮 A+B 被按下时**<br>（黄色和蓝色按钮） | 滚动Team A 和 Team B 的分数。 |
| **当振动** | 通过点亮 **绿色LED**（持续30秒）然后点亮 **黄色LED**（持续20秒）最后点亮 **红色LED**（持续10秒）<br>来启动计时器1分钟。时间到时播放旋律 “wawawawaa”。<br>切换**红色LED** 10次。 |

这是给你的提示。你需要制作两个变量，分别命名为 Team A 和 Team B。

# 让我们一起玩

Win, Lose or Draw~


## 怎么玩 ：

- 将课程分为2组 - Team A 和 Team B。
- A 团队的一名成员将首先随机挑选一张牌。静默读取卡上的单词后，摇动 EDU:BIT 以启动计时器（1分钟）。
- 然后，他或他就可以开始在板上画画，供团队成员猜测。禁止讲话或打手势！
- 如果在时间到了之前，任何团队成员正确猜出了单词或短语，则对团队A奖励1分（按按钮A或黄色按钮）。
- 如果团队A 失败，则团队B 可以通过给出最佳猜测来尝试 “窃取” 分数。
- 两支球队轮流抽签并猜测，直到比赛结束。
- 得分最高的团队获胜！

**注意：**

请扫描此处以下载带有建议质询字词的可打印卡片。 如果你不喜欢绘图，可以尝试 Charades。 遵循相同的规则，但你可以使用手势来示意线索，以供队友猜测，而不是绘制。 希望你玩得开心！

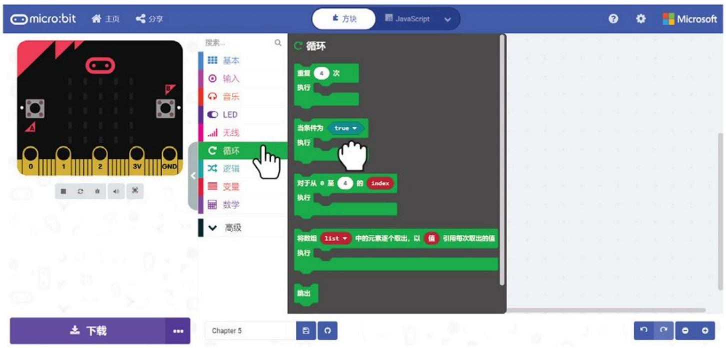
第5章

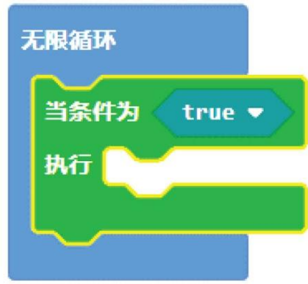
# IR数字骰子。让我们掷骰子

(IR Bit)

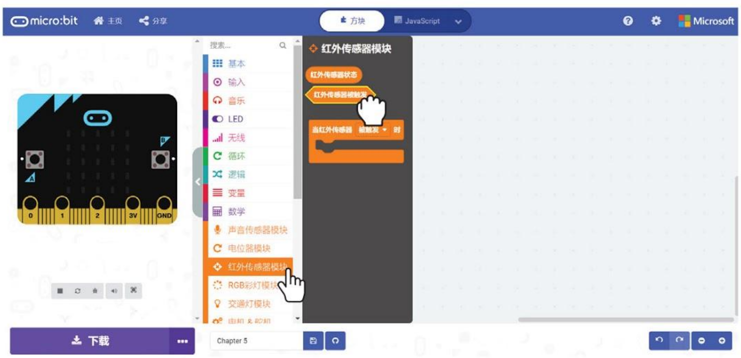
## 让我们编码！

第1步 在 MakeCode 编辑器中创建一个新项目，并添加 EDU:BIT 扩展名(你可以参考第40页)。单击 [循环] 类别，然后选择 [当条件为 __ 执行] 块。将块对齐到 [无限循环] 插槽。

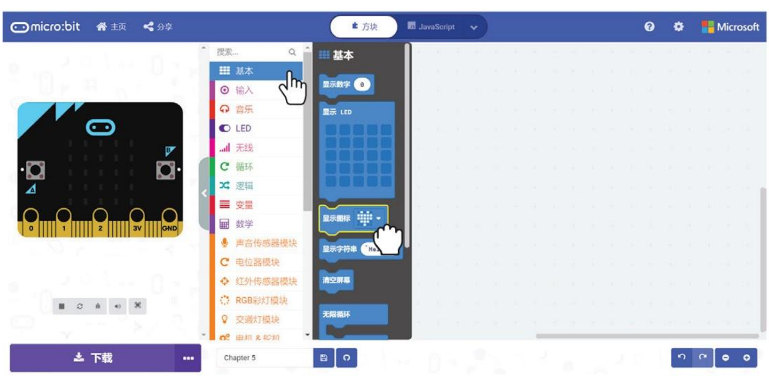


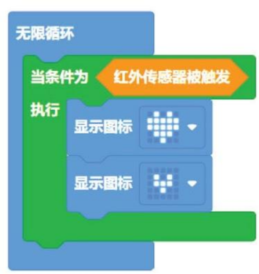



第2步 单击 [红外传感器模块] 类别，然后选择 [红外传感器被触发] 块。将块对齐到 [当条件为 __ 执行] 块上的条件槽。

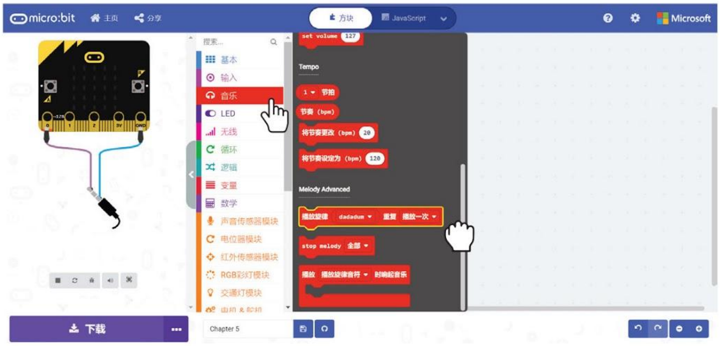


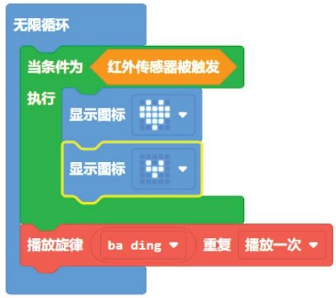

第3步 单击 [基本] 类别，然后添加两个 [显示图标] 块。将图标之一更改为“小心脏”。将两个块都对齐到 [当条件为 __ 执行] 块。





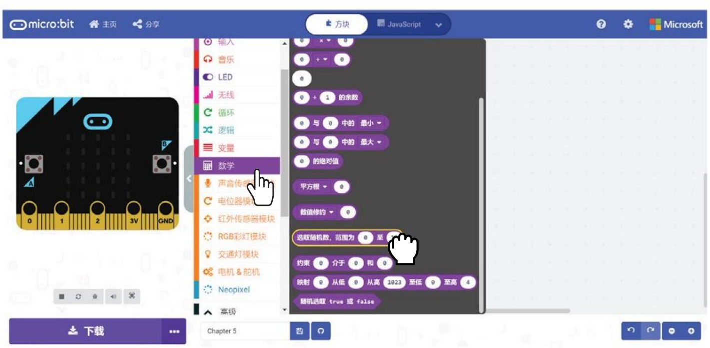
第4步 单击 [音乐] 类别，然后选择 [播放旋律 __ 重复 __ ] 块。将旋律“dadadum” 更改为 “**ba ding**”。

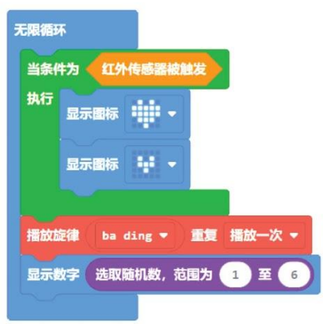





第5步 单击 [基本] 类别，然后选择 [显示数字] 块。


第6步 单击 [数学] 类别，然后选择 [选取随机数，范围为 __ 至 __ ] 块。将数字设置为 **1** 和 **6** 。





第7步 单击 [循环] 类别，然后选择 [当条件为 __ 执行] 块。

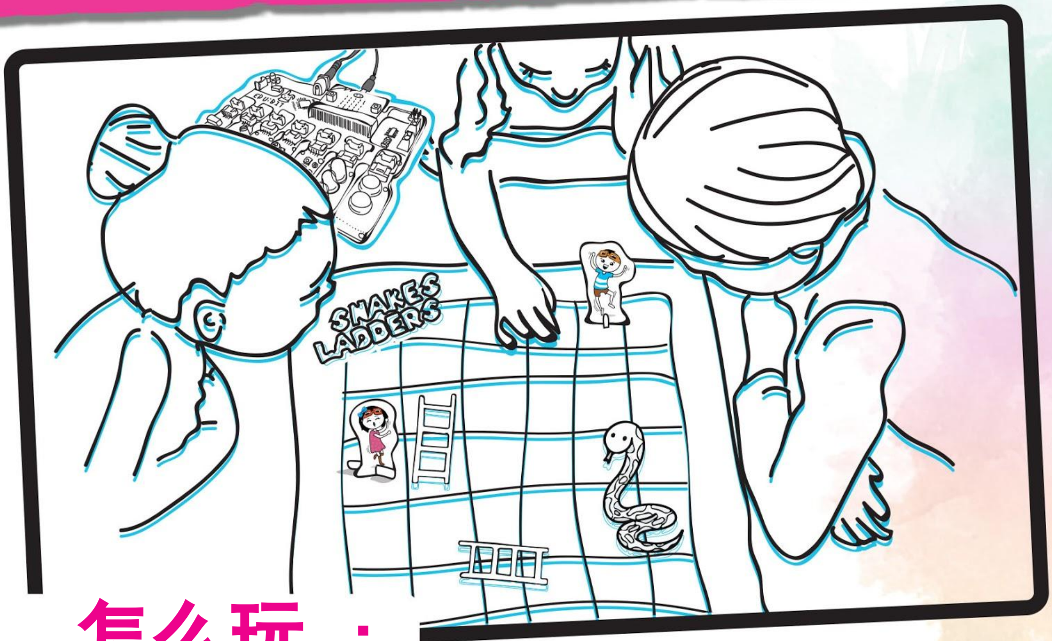
第8步 单击 [逻辑] 类别，然后选择布尔值 [非 __ ] 块。将块对齐到 [当条件为 __ 执行] 块上的条件槽。

第9步 单击 [红外传感器模块] 类别，然后选择 [红外传感器被触发] 块。将块对齐到 [非 __ ] 块的空白插槽。

第10步 将代码刷新到你的 EDU:BIT。

# 让我们一起玩

## 蛇和梯子游戏




## 怎么玩 ：

- 每个玩家选择一个角色并将其放置在显示 “Start Here” 的空间上。
- 然后，玩家轮流 “掷骰子” - 将手掌放在 IR Bit 上方。当你看到心脏跳动的动画时，请移开手掌。
- 将你的角色向前移动 LED 矩阵上显示的空格数（介于 1 到 6 之间）。
- 如果你的角色落在梯子的底部，则可以向上移动到梯子的顶部。如果你的角色落在蛇的头上，则必须向下滑动至蛇尾的尖端。
- 首位达到 100 位的玩家就是赢家。希望你玩得开心！

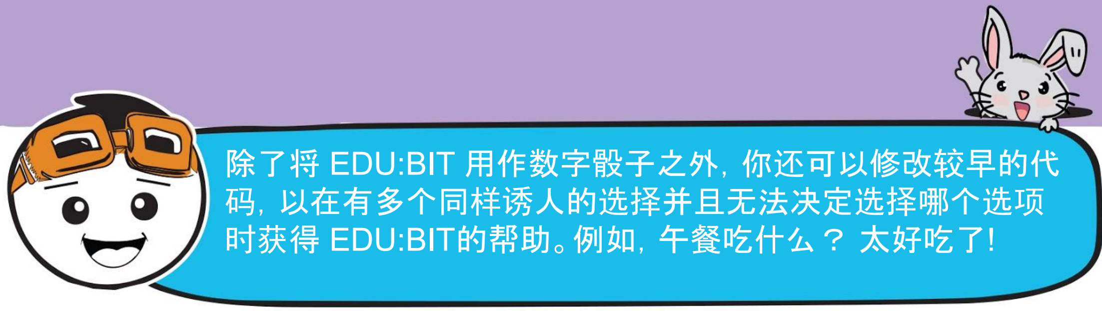
**注意！**

盒子中提供了蛇和梯子游戏板和角色块。弹出角色及其底部的支架，然后将它们组合在一起以形成角色。

# 破解密码

我们之前在代码中使用了 **[循环]** 类别中的 **[当条件为 __ 执行]** 块。你知道 while loop 如何运行吗？

当程序进入 **[当条件为 __ 执行]** 块时，它将检查条件。满足条件（或为TRUE）时，程序将执行 **[当条件为 __ 执行]** 块中的代码块。它将继续循环，但是一旦不满足条件（或变为FALSE），程序将退出 while loop 并运行下一个代码块。




第11步 单击 **[高级]：[数组]** 类别，然后选择 **[将 text list 设为 数组 _ _ _ ]** 块。将块对齐到 **[基本]：[当开机时]** 插槽。


第12步 单击 **[text list]** 块，然后选择 “**重命名变量**”。在弹出窗口中键入 “**午餐选择**”，然后单击 “确定”。



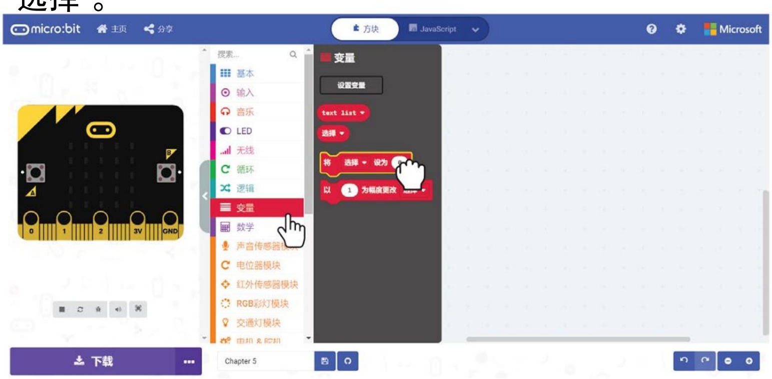
第13步 单击数组块，然后在每个块中键入一个午餐选项。


单击此按钮添加更多选项。

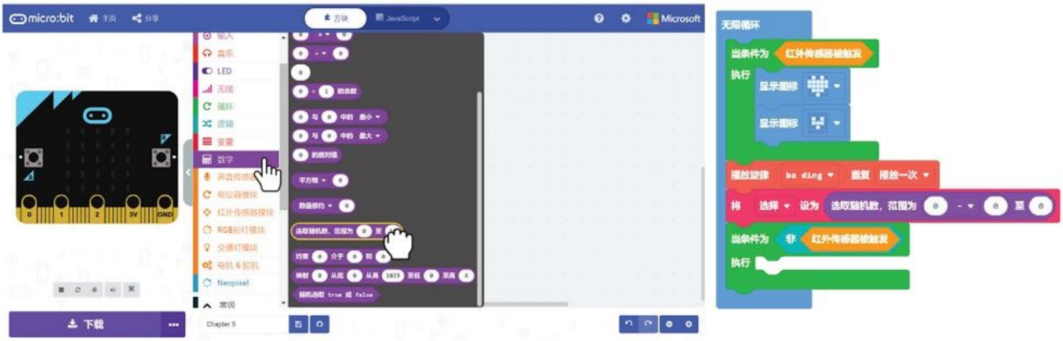
第14步 单击［变量］类别，然后创建一个名为“选择”的新变量。


第15步 右键单击［显示数字［选取随机数，范围为 __ 至 __］］块，然后选择“**删除块**”。

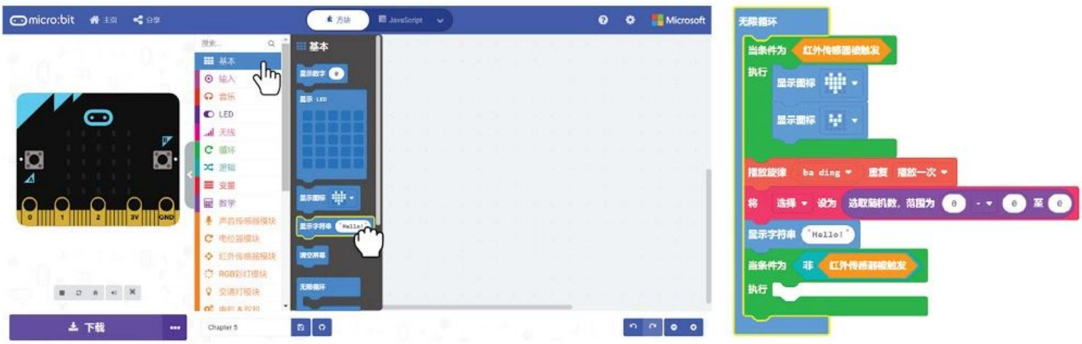

第16步 单击［变量］类别，然后选择［将 __ 设为 __］块。将块放置在［播放旋律 __ 重复 __］块和［当条件为 __ 执行］块之间。将变量更改为“选择”。



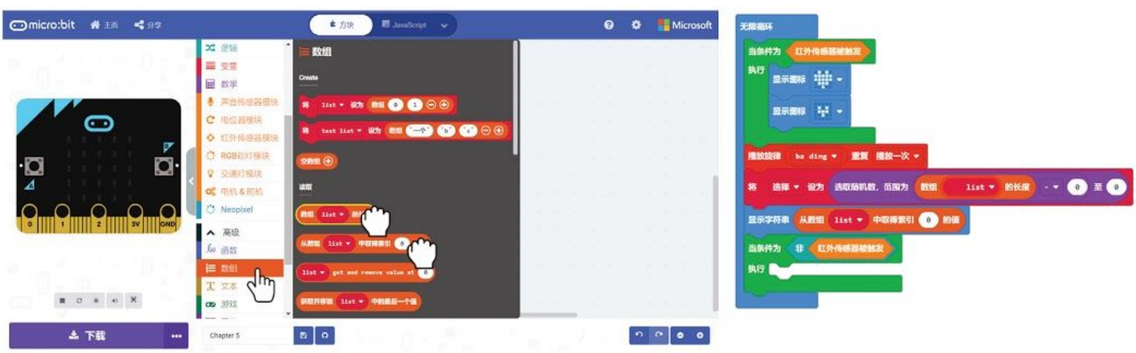

第17步 单击 **[数学]** 类别，然后将 **[选取随机数，范围为 __ 至 __ ]** 和 **[ __ - __ ]** 块添加到你的代码中。将最后一个值更改为**1**。



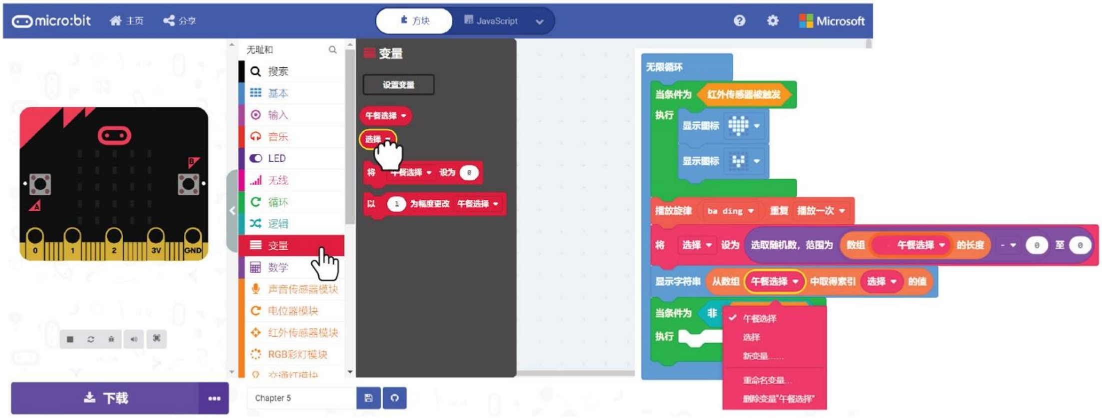
第18步 单击 **[基本]** 类别，然后选择 **[显示字符串]** 块。



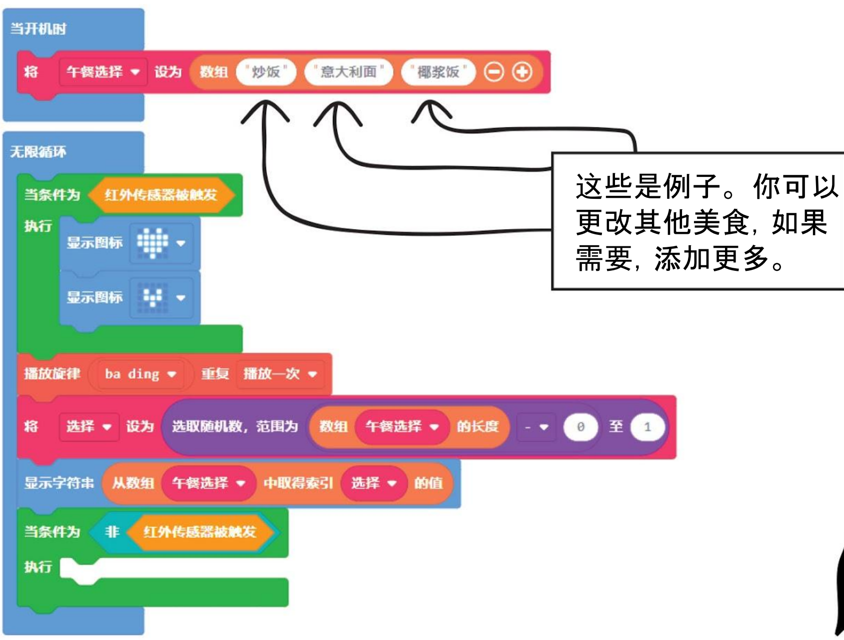
第19步 单击 **[高级]** : **[数组]** 类别，然后将 **[数组 __ 的长度 ]** 和 **[从数组 __ 中取得索引 __ 的值]** 块添加到你的代码中。



第20步 单击 **[list]**，然后将两个块的变量都更改为“**午餐选择**”。最后，单击 **[变量]** 类别，然后选择 **[选择]** 块。将其对齐到 **[从数组 __ 中取得索引 __ 的值]** 块的空白插槽。




这是你完整的“午餐是什么”编码：



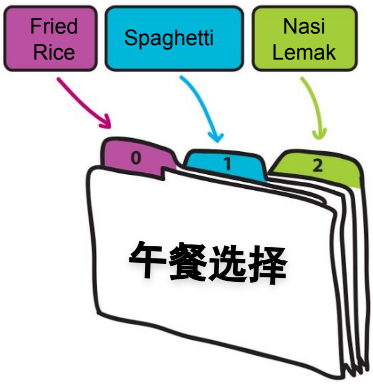

下次不确定吃什么时，可以通过触发 IR BIT，然后将手移开，让 EDU:BIT 为你决定。你还可以修改此代码，以帮助你决定与朋友一起玩哪个游戏。你知道要做哪些更改吗？

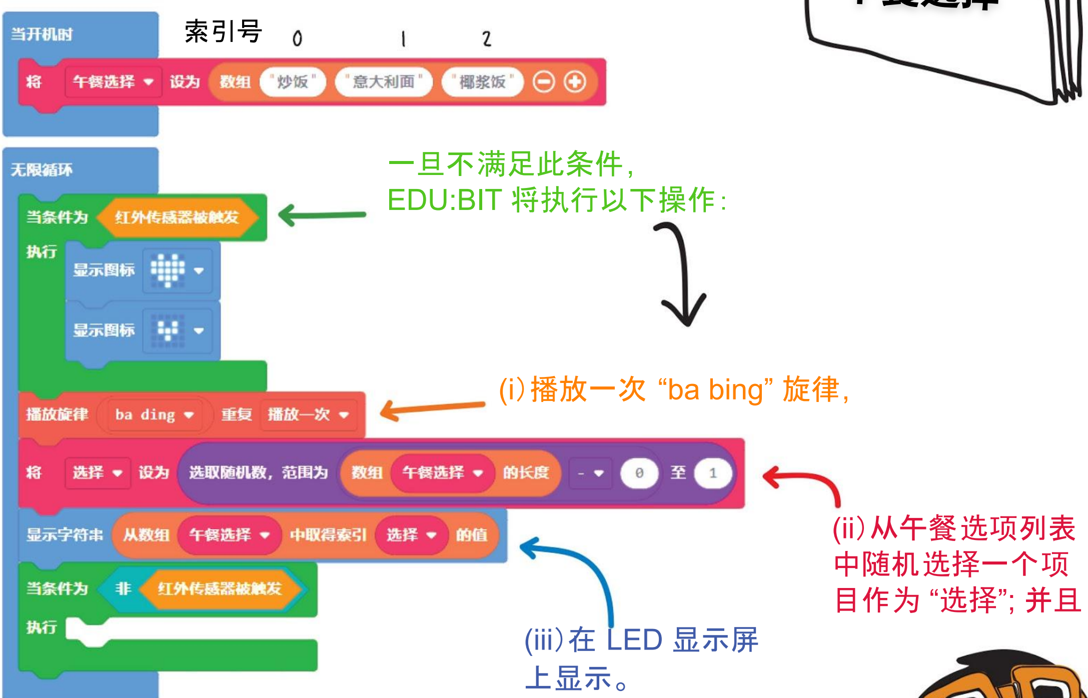
# 破解密码

**数组 (array)** 是相关变量的列表或集合。你可以将其视为包含多个部分的文件夹，每个部分用于存储一条信息。我们使用数组，以便在需要从列表中添加或删除元素时可以轻松地修改代码。

例如，在这段代码中，我们创建了一个包含三个元素的数组，并将其命名为 “午餐选择”。然后，我们可以轻松地编辑每个元素代表的食品。

我们还可以通过单击 “+” 或 “-” 按钮来添加选项或减少列表中的元素数量。

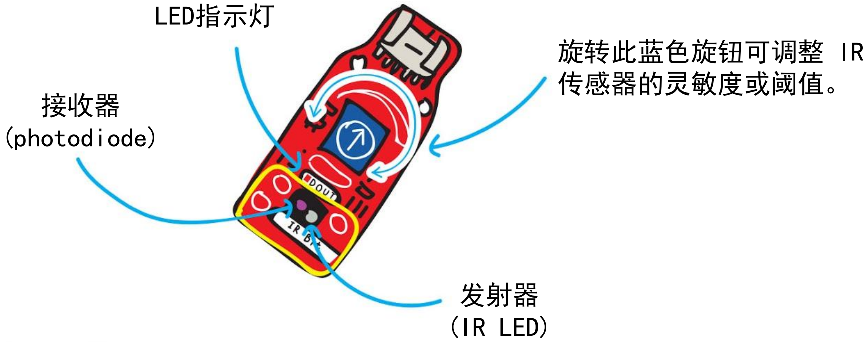



在编程中，我们从 0 开始而不是从 1 开始计数。因此，“fried rice” 在列表中的索引号为 0，而最后一个元素 “nasi lemak” 在索引号 2，即使它是第三项。

# 有趣的事实！

**红外(IR)传感器** 是一种电子仪器，通常用于检测障碍物。 红外传感器由两部分组成 – 发射器（IR LED）和接收器（photodiode）。

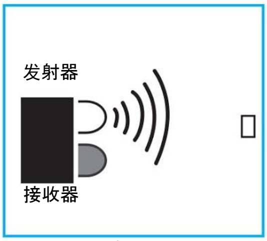



## 它如何运作？

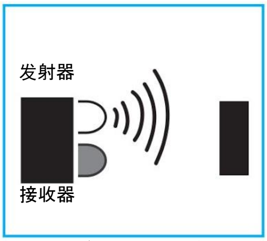
如果将物体放在传感器前面，IR LED 会发出红外光，该红外光会反射到接收器。 如果反射光量大于阈值，则 IR BIT 将被“触发”。 触发后，IR BIT 上的 LED 指示灯将点亮。

如果没有物体或物体离得太远，则很少或没有红外光会反射到接收器。 因此，IR BIT 将不会被触发。

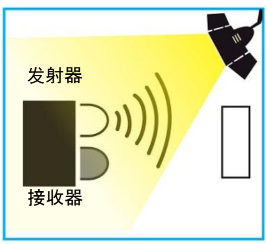
但是，在以下情况下，红外传感器可能无法正常工作：



对象太小



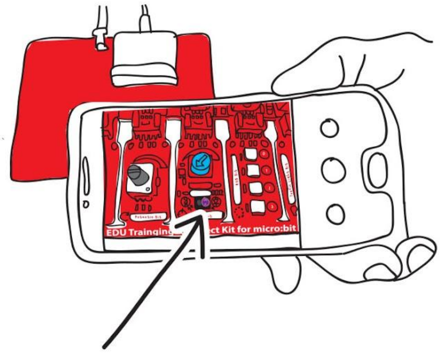
对象的表面为黑色或深色



光干扰

## 你知道吗？

肉眼看不到红外光，但是，你可以通过手机摄像头查看红外 LED 来查看红外光。

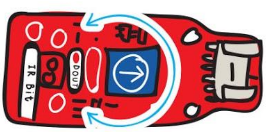


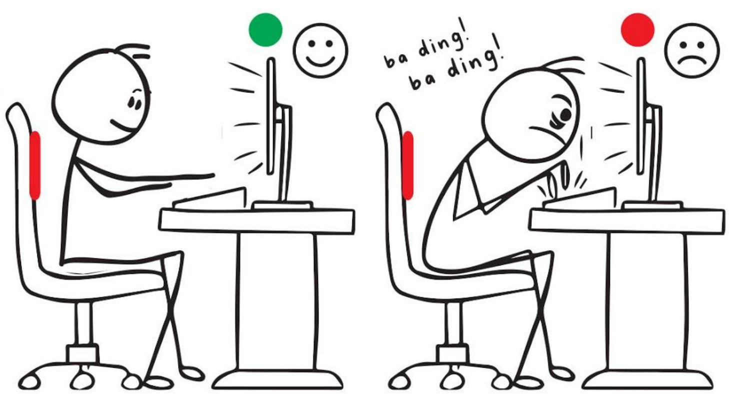
# 编码挑战

| 编写 EDU:BIT 程序以用作防松弛检测器。 | |
|---|---|
| **当开机时** | 滚动提醒 – “Mind your posture.” |
| **红外传感器被触发时** | 在 LED 矩阵显示屏上显示笑脸并点亮绿色 LED。 |
| **当红外传感器未被触发时** | 开始重复“ba ding”旋律。 在 LED 矩阵显示屏上显示悲伤的脸，并使红色 LED 闪烁。 |

## 运作方式：

如图所示，将 EDU:BIT 附加到椅子的背面。以良好的姿势舒适地坐下。调整 IR BIT 上的蓝色旋钮，直到 LED 指示灯点亮（IR BIT 检测到你的背部）。此过程称为校准。



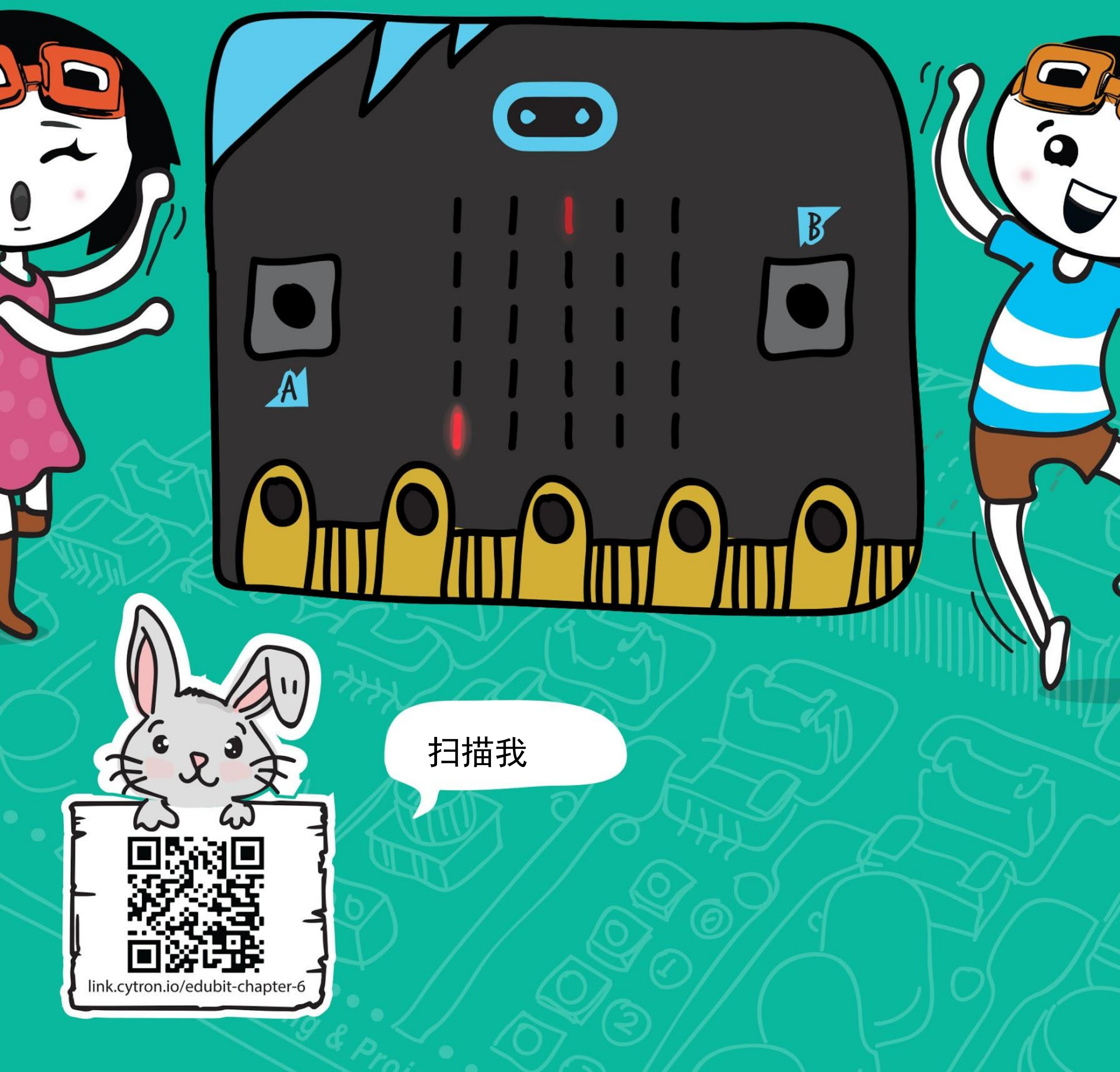




如果你穿着其他颜色的衬衫，则需要重新校准 IR Bit。 你知道为什么吗？

# 第6章

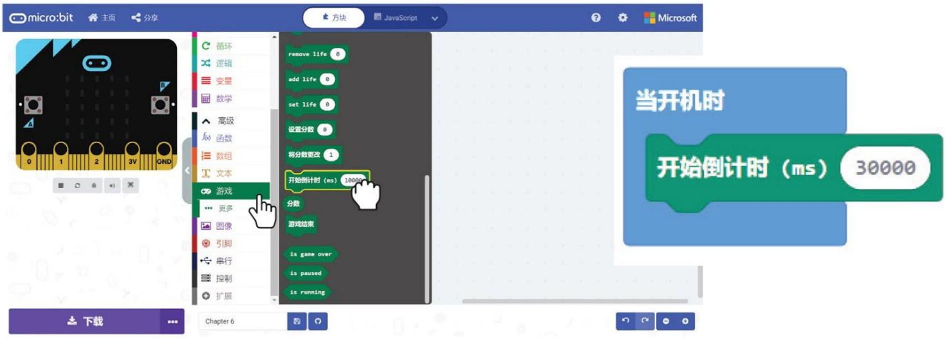
# 捉人游戏

(Potentio Bit)

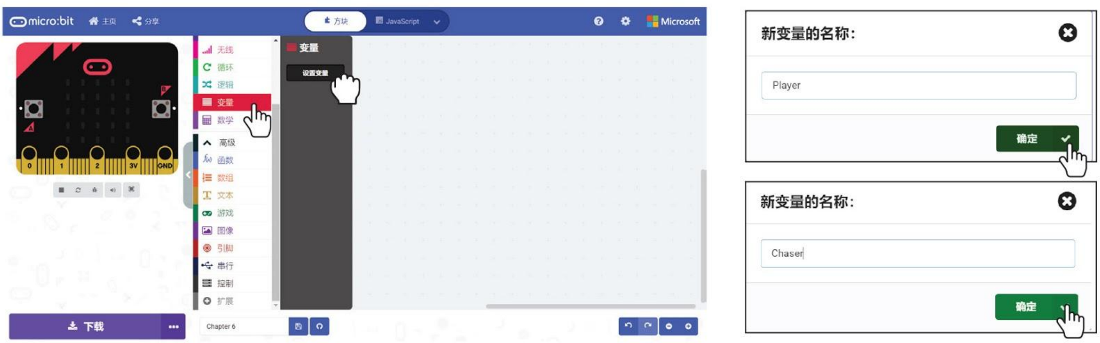


# 让我们编码！

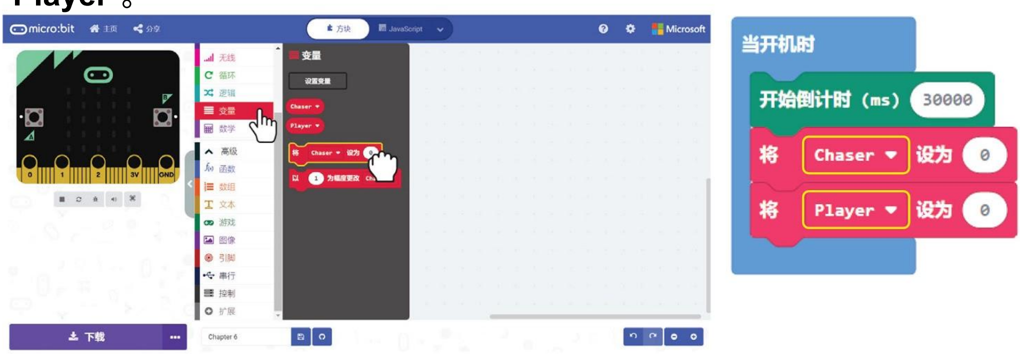
第1步 在你的 MakeCode 编辑器中，创建一个新项目并添加 EDU:BIT扩展名（你可以参考第40页）。单击 **[高级]**，然后单击 **[游戏]** 类别。选择 **[开始倒计时(ms)__]** 块，将其捕捉到 **[当开机时]** 块并将其值更改为 **30000**。




第2步 单击 **[变量]** 类别，然后单击 **[设置变量]**。在弹出窗口中键入 “**Player**”，然后单击 “确定”。创建另一个名为 “**Chaser**” 的变量。

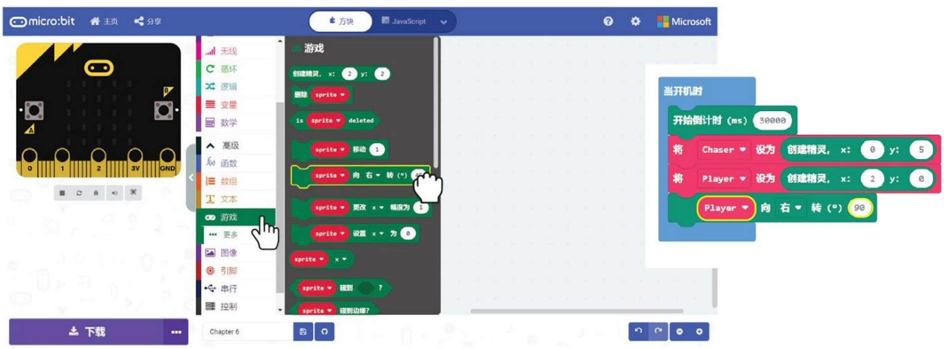



第3步 单击 **[变量]** 类别，然后选择 **[将 __ 设为 __]** 块。复制 **[将 __ 设为 __]** 块，并将两个块都捕捉到 **[当开机时]** 块。将变量分别设置为 “**Chaser**” 和 “**Player**”。

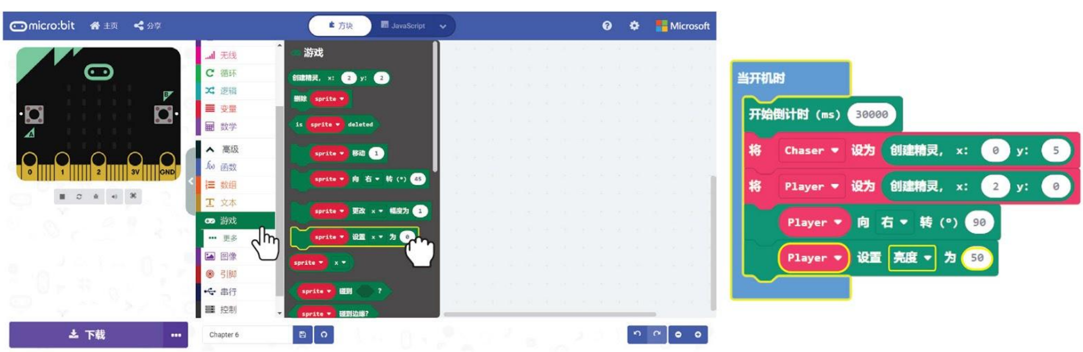


第4步 单击 **[高级]**，然后单击 **[游戏]** 类别。选择 **[创建精灵 x: __ y: __]** 块。将这些块复制并对齐到 **[将 __ 设为 __]** 块。将 “**Chaser**” 的值更改为x : **0** y : **5**，将 “**Player**” 的值更改为 x : **2** y : **0** 。


第5步 单击 **[高级]** : **[游戏]** 类别，然后选择 **[ __ 向 __ 转(°)__ ]** 块。将块放置在 **[当开机时]** 插槽中。选择变量 “**Player**” 并将度数设置为 **90**。

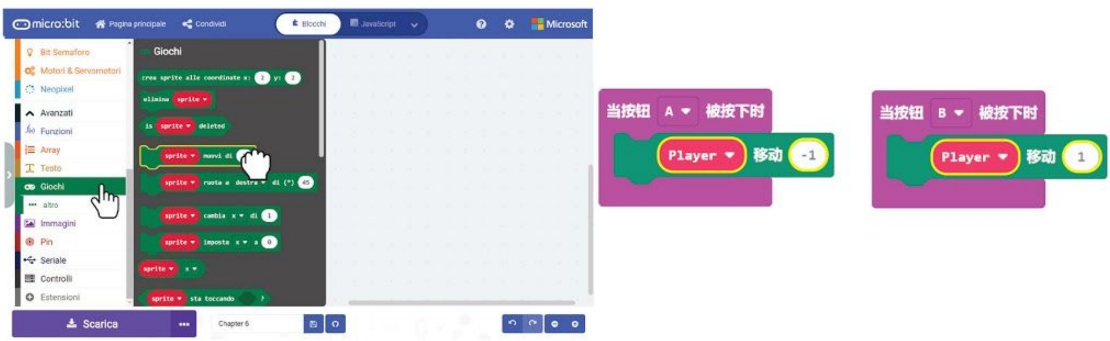


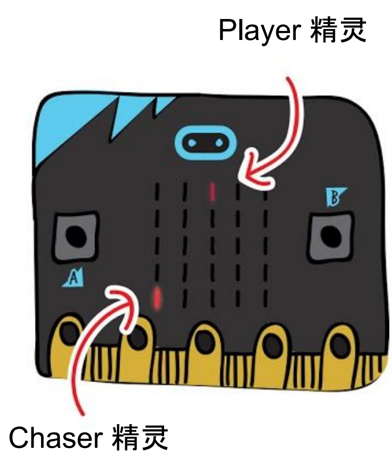

第6步 单击 **[高级]** : **[游戏]** 类别，然后选择 **[ __ 设置 __ 为__ ]** 块。选择变量 “**Player**”，将 “**x**” 更改为 “**亮度**”，并将亮度设置为 **50**。



第7步 单击［输入］类别，然后选择［当按钮 __ 被按下时］块。复制该块，然后在第二个［当按钮 __ 被按下时］块上选择按钮 “**B**”。


第8步 单击 **[游戏]** 类别，然后选择 **[ __移动 __ ]** 块。复制这些块并将其放置在［当按钮 __ 被按下时］插槽中。为两个块选择变 “**Player**”，并将值更改为 “**-1**”（按下按钮A）和 “**1**”（按下按钮B）。



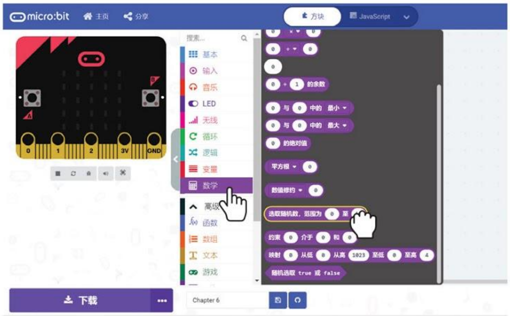




将代码刷新到你的 EDU:BIT。当你按下蓝色按钮（按钮B）时，你是否注意到调光器 LED 灯向下移动？
那是 Player 精灵！精灵就像你可以控制的 “小小的 LED 生物”。

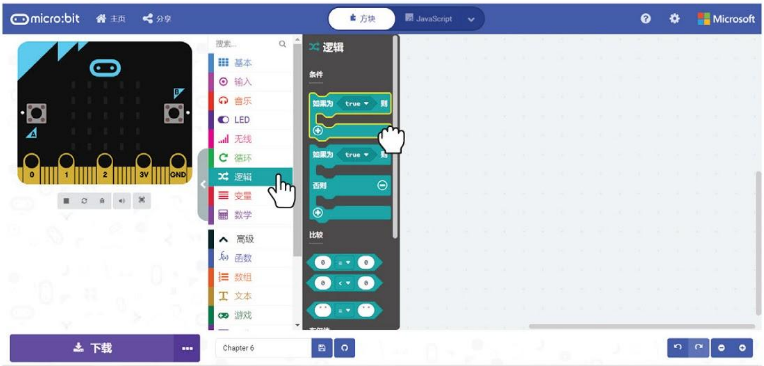
如果你按下黄色按钮（按钮A）会怎样？

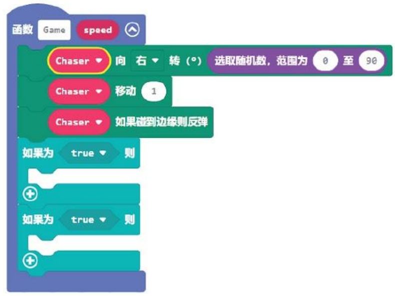
第9步 单击 **[高级]** 类别，然后选择 **[函数]** 类别，然后单击 **[创建一个函数]**。在“编辑函数”窗口中，将“doSomething”重命名为“**Game**”。接下来，单击 **[数字]** 添加一个参数，然后在功能块中将‘数字’重命名为 **'speed'**。然后点击“完成”。


第10步 通过添加 **[高级]**：**[游戏]** 和 **[数学]** 类别中的块来继续构建代码，如下所示。请记住将变量更改为“**Chaser**”，并将其值更改为 **90**。

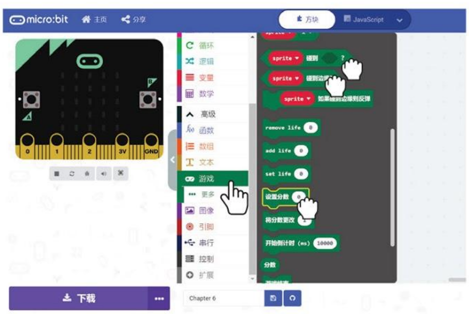



**第11步** 将来自［逻辑］类别的两个［如果-则］块添加到你的代码中。






**第12步** 通过添加 **[高级]**：**[游戏]** 和 ［**数学**］ 类别中的块来继续构建代码，如下所示。




第13步 通过单击将两个［sprite］块更改为“ Chaser”。 单击［变量］类别，然后选择［Player］块。 将其对齐到［__ 碰到 __］块的空白插槽。


第14步 单击［基本］类别，然后选择［暂停（ms） __ ］块。 添加到你的编程代码。 单击功能块上的［speed］，按住并拖动到［暂停（ms） __ ］块的空白位置。



当“Chaser”触摸“Player”时，你可以通过播放旋律来增加游戏的趣味性。 你能找出要添加的块以及在何处放置块吗？

现在，让我们为游戏添加不同的难度级别！

第15步 单击［逻辑］类别，然后选择［如果-则-否则］块。将块放置在［无限循环］插槽中。单击（+）按钮添加一个语句。

第16步 单击［函数］类别，然后选择［调用 Game］。复制该块，然后将块捕捉到每个［如果-则-否则］插槽。相应地将［调用 Game］块的值更改为250、500 和 750。

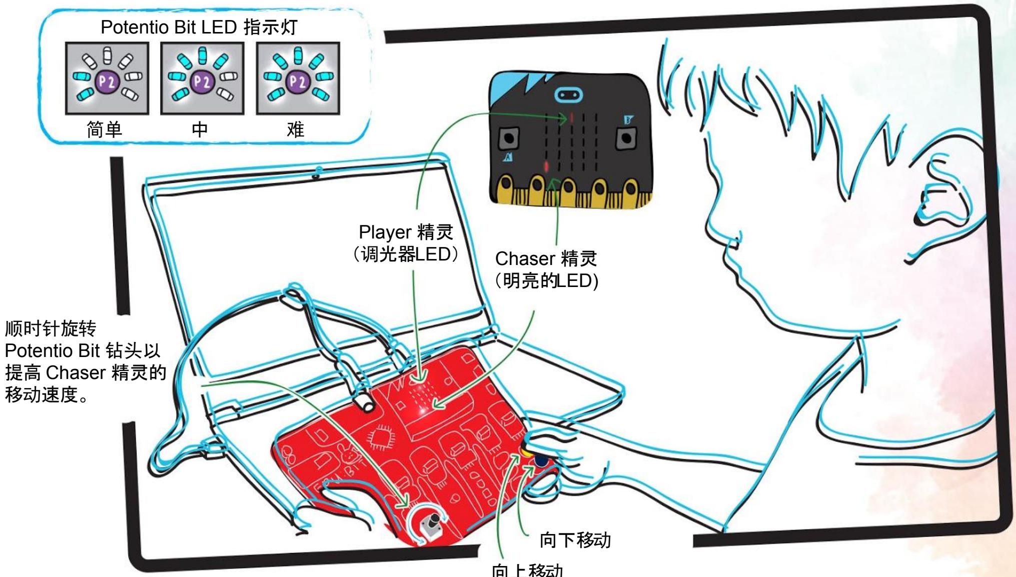
第17步 单击［电位器模块］类别，然后选择［电位器值> __ ］块。复制这些块并将其放置在［如果-则-否则］块的条件槽中。将第一个块的值设置为 800，将第二个块的值设置为 400。

这是完整的编程代码：


第18步 将完整的代码上传到你的 EDU:BIT，并与你的朋友一起玩捉人游戏。

# 让我们一起玩

Tag， You're It!




## 怎么玩 ：

通电后，Chaser 精灵将继续沿随机方向移动。

上下移动 Player 精灵，以避免 Chaser 精灵。按下黄色按钮（按钮A）向上移动，或按下蓝色按钮（按钮B）向下移动。

如果 Chaser 精灵“触摸”了 Player 精灵，或者在30秒计时器计时结束后，游戏结束。

每次 Chaser 精灵接触边缘时，您将获得1分。得分最高的玩家是获胜者。玩得开心~

### 提示：

#1 为了在30秒的时间限制内获得更高的分数，可以提高 Chaser 精灵的速度，使其更频繁地“接触”边缘。
#2 游戏结束后，你可以同时按按钮A + B来开始新游戏。这是 **[Game]** 块的内置功能。

# 破解密码

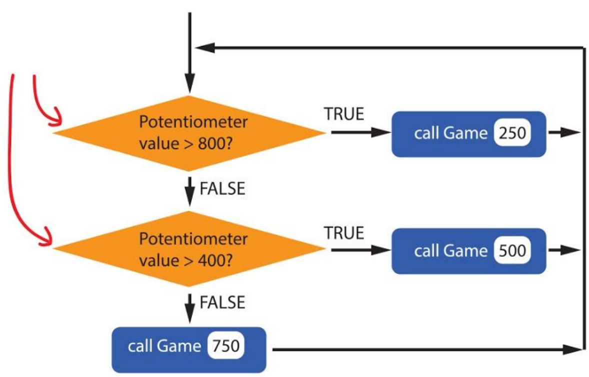
在编程中，我们使用条件 if 语句进行决策。在 MakeCode 中，我们使用**［逻辑］**类别中的**［如果–则］**或**［如果–则–否则］**条件块形成条件。程序检查条件语句，如果条件为 TRUE（满足此条），则执行条件块中的代码。否则，如果 FALSE，它将继续进行下一个代码块。


**如果**满足此条件（Chaser 精灵触摸 Player 精灵），

**则**执行此操作（播放旋律 wawawawaa，然后显示“游戏结束”动画）。

当我们有多个条件时，程序将从上到下依次评估条件，并执行返回 TRUE的第一个条件的相应代码。因此，与以下条件相比，条件越高，其拥有的优先级越高。

例如，游戏中的此代码通过将电位计的值与预设阈值进行比较来确定 Chaser 精灵的移动速度。


**如果**电位计值 > 800，则调用函数 Game（变量 speed 设置为 250 ms），**否则**，如果电位计值 > 400，则调用函数 Game（speed = 500ms），**否则**调用函数 Game（speed = 750ms）。

# 探索更多编码块

**#1** 将 **[基本] ： [显示数字]** 块与 **[电位器模块] ： [电位器值]** 块一起使用来读取并显示当前的电位器值。


**#2** 电位计返回0到 1023 之间的值。你可以使用 **[高级]:[引脚]** 类别中的 **[ 映射 __ 从低 __ 从高 __ 至低 __ 至高 __ ]** 块将读数映射到更有意义的另一个范围。

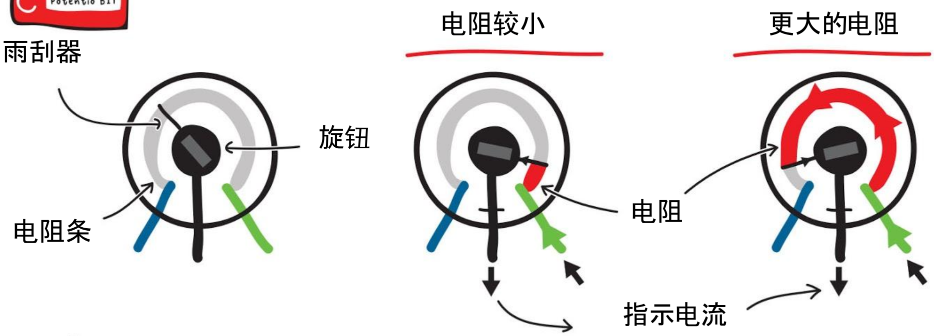

**#3** 映射块将返回带小数点的数字（例如1.68、3.998）。要四舍五入，请使用 **[数学]** 类别中的 **[ 数值修约__ ]** 块。

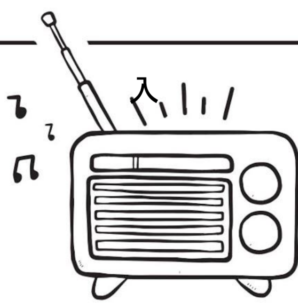

这是将 Potentio Bit 读数映射到 0 到 7 范围内的示例代码。该值将四舍五入并显示在 LED 矩阵上。


# 有趣的事实！


**Potentio Bit 电位计**，也称为“ pots”，是可变电阻，其电阻可以使用旋钮或滑块轻松调节。

如果你有一个 10,000Ω 的电位计，则可以通过更改抽头的位置来获得介于 0Ω 至 10,000Ω 之间的电阻值。



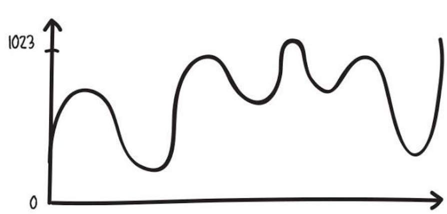



**以下是一些常见的应用程序：**

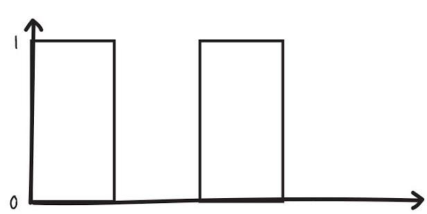
- 扬声器的音量
- 收音机的频率控制
- 热水器温度控制

EDU:BIT上的电位计是一种模拟输入设备。它测量电势并将所测量的电压(0V 至 3.3V 之间)转换为 0 至 1023 之间的整数值。

模拟信号



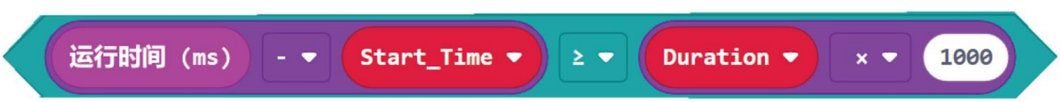

数字信号



# 编码挑战

| | |
|---|---|
| 将 EDU:BIT 编程为计时器。用户 Potentio Bit 调整持续时间(0 到 60 秒之间),按钮A 激活计时器,按钮B 重置。 | |
| **当开机时** | 将 Mode 设置为 0 |
| **当按钮 A 被按下时**<br>(黄色按钮) | 将 Mode 设置为1<br>将 Start Time 设置为运行时间<br>显示笑脸图标 |
| **当按钮 B 被按下时**<br>(蓝色按钮) | 将 Mode 设置为 0 |
| **无限循环** | 始终检查模式<br>• 如果 Mode = 0,则将 Duration设置为电位计读数的汇总值,映射到低 0 和高 60,并在 LED 矩阵上显示 Duration。<br><br>• 否则,如果 Mode = 1,请检(运行时间 – Start Time) > (Duration x 1000)。如果 TRUE,请播放旋律 wawawawaa,然后将 Mode 设置为 2。<br><br>• 否则显示悲伤的脸图标。 |

这里有一些提示给你:
提示1 你需要创建三个变量:Mode,Start Time 和 Duration。
提示2 **[ 运行时间 (ms) __ ]** 块来自 **[输入]** 类别。
提示3 使用此条件语句来检查时间是否到了。



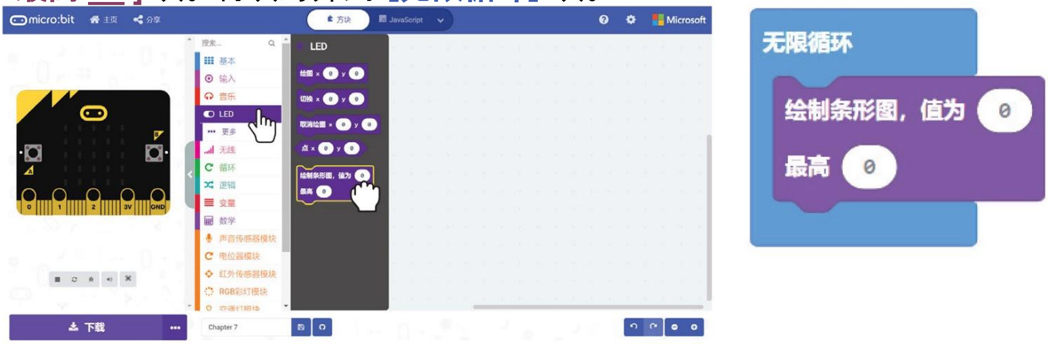
# 第7章

# 听掌声响起！

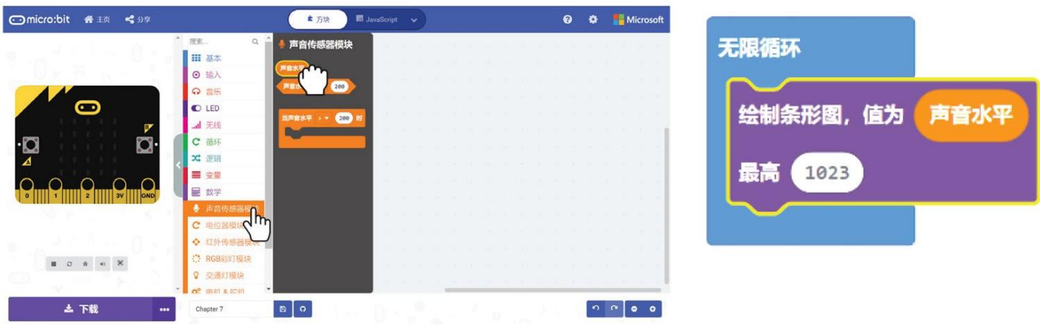
(Sound Bit)

## 让我们编码！

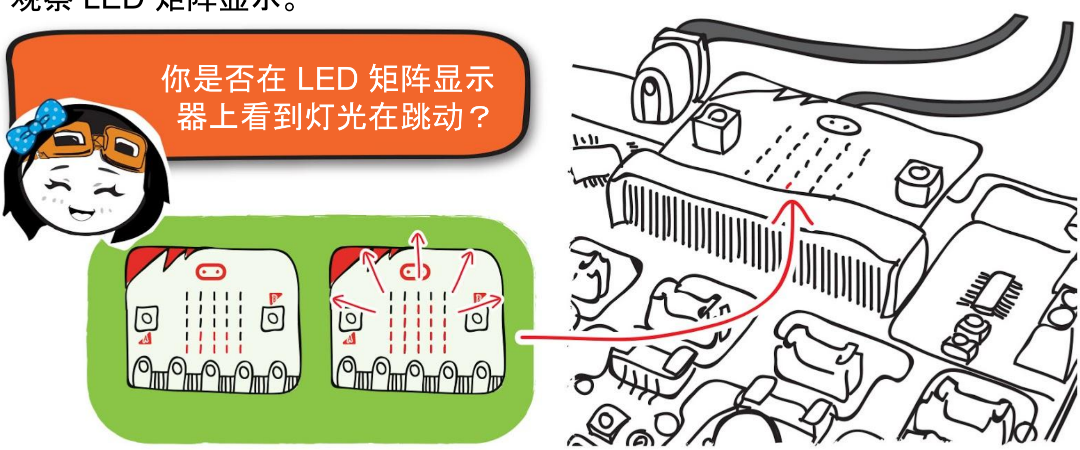
第1步 在 MakeCode 编辑器中创建一个新项目，并添加 EDU:BIT 扩展名 (你可以参考第40页)。单击 **[Led]** 类别，然后选择 **[绘制条形图，值为 __ 最高 __ ]** 块。将块对齐到 **[无限循环]** 块。




第2步 单击 **[声音传感器模块]** 类别，然后选择 **[声音水平]** 块。将块对齐到 **[绘制条形图，值为 __ 最高 __ ]** 块，并将秒 0 的值更改为**1023**。




第3步 将代码刷新到你的 EDU:BIT。当你拍手或在桌子上点击手指时，请观察 LED 矩阵显示。



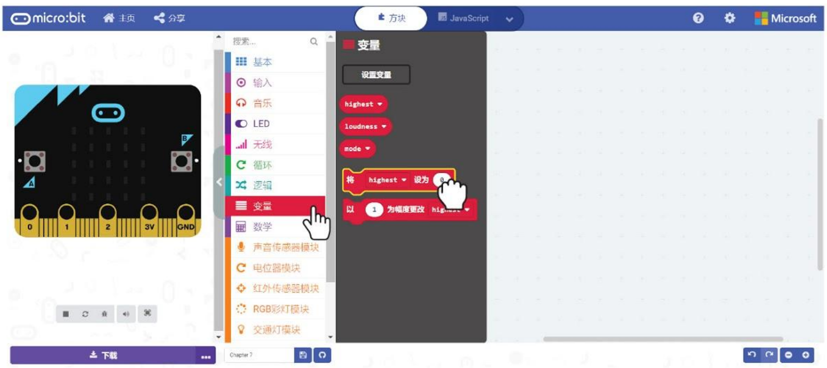
现在让我们将 EDU:BIT 变成掌声响的探测器。

第4步 开始一个新项目并添加 EDU:BIT 扩展名。单击 **[变量]** 类别，然后单击 **[设置变量]**。在弹出的窗口中键入“**mode**”，然后单击确定。再创建两个名为 “**loudness**” 和 “**highest**” 的变量。

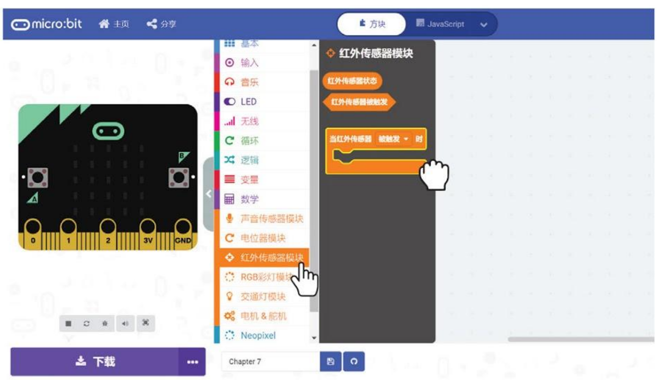

第5步 从 **[变量]** 类别中获取 **[将 __ 设为 __]** 块。将块卡入 **[当开机时]** 插槽。将变量设置为 “**mode**”。


第6步 单击 **[输入]**，然后选择 **[当按钮 __ 被按下时]** 块。

第7步 从 [变量] 类别中获取两个 [将 __ 设为 __] 块，然后将这些块对齐到 [当按钮 A 被按下时] 插槽。将第一个变量设置为 “**mode**”，将值设置为 1，将第二个变量设置为 “**highest**”，默认值为 **0**。

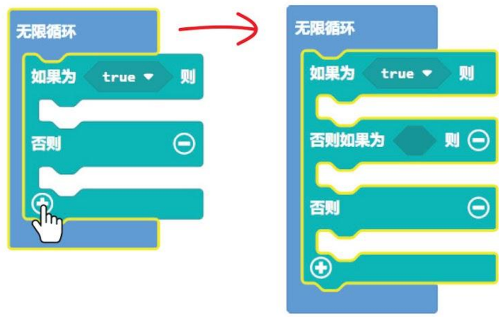



第8步 单击 [红外传感器模块] 类别，然后选择 [红外传感器被触发时] 块。

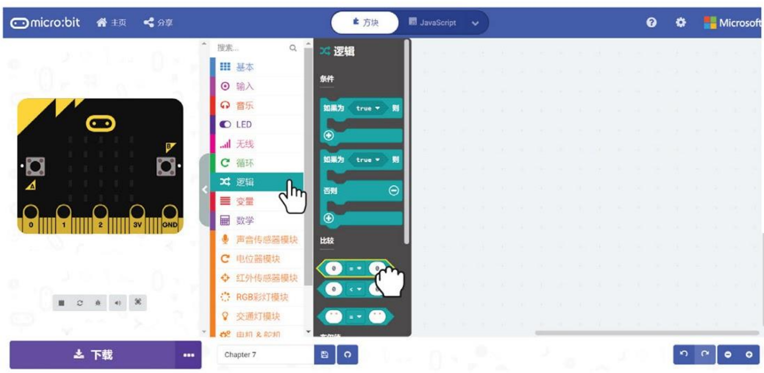


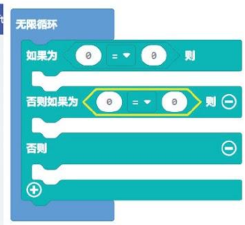

第9步 单击 [变量] 类别，然后选择 [将 __ 设为 __] 块。将模块吸附到 [红外传感器被触发时] 插槽，并将变量更改为 “**mode**”，并将其值更改为 **2**。

第10步 单击［逻辑］类别，然后选择［如果–则–否则］块。将模块卡入［无限循环］插槽。单击毛绒图标，将另一个否则条件添加到块中。

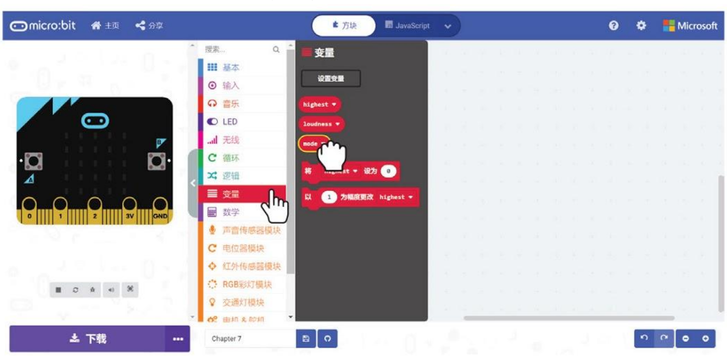



第11步 单击［逻辑］类别，然后选择［ __ = __ ］比较块。复制该块，然后将块对齐到［如果–则–否则］块的条件槽。

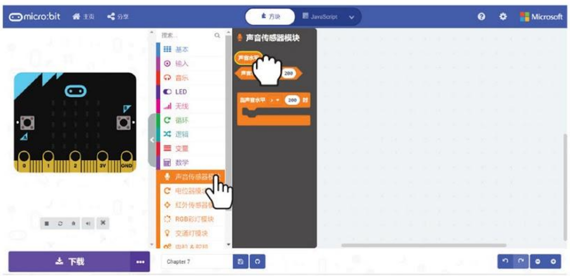




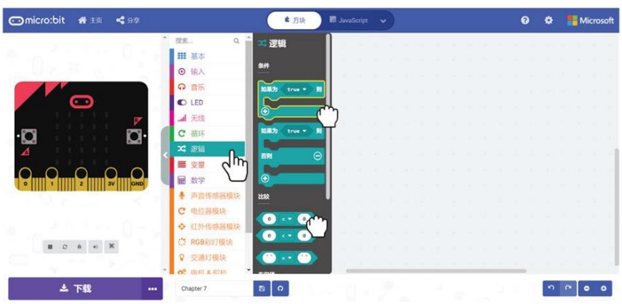

第12步 从［变量］类别中获取［mode］，并将其捕捉到比较块的左侧插槽中。将其他插槽分别设置为 1 和 2。



第13步 从 [变量] 类别中获取 [将 __ 设为 __] 块，然后将该块对齐到第一个 [如果-则-否则] 插槽。将变量设置为 “loudness”，然后将 [声音水平] 类别从 [声音传感器模块] 类别捕捉到值插槽。

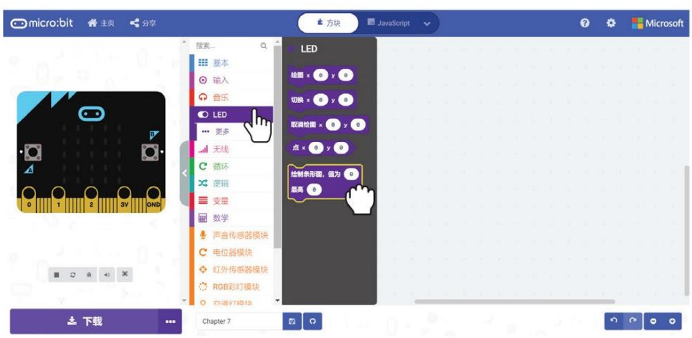


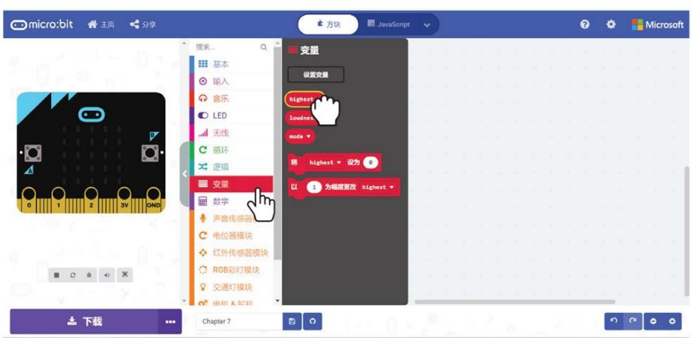

第14步 单击 [逻辑] 类别，然后选择 [如果-则] 块和 [ __ = __ ] 比较块。将符号=更改为 > 。从 [变量] 类别中将 [loudness] 和 [highest] 捕捉到比较块的插槽中。



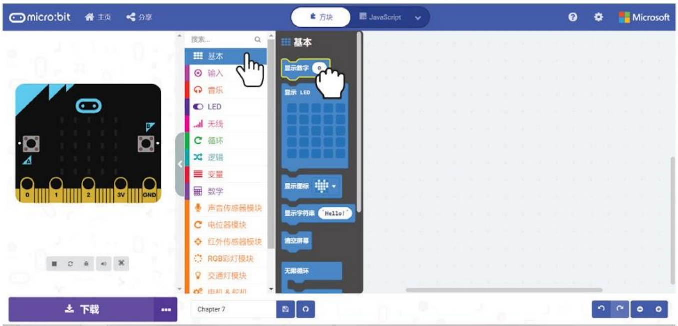
第15步 单击 [变量] 类别，然后选择 [将 __ 设为 __] 块。将块捕捉到 [如果–则] 插槽，将 [loudness] 从 [变量] 类别捕捉到值插槽。

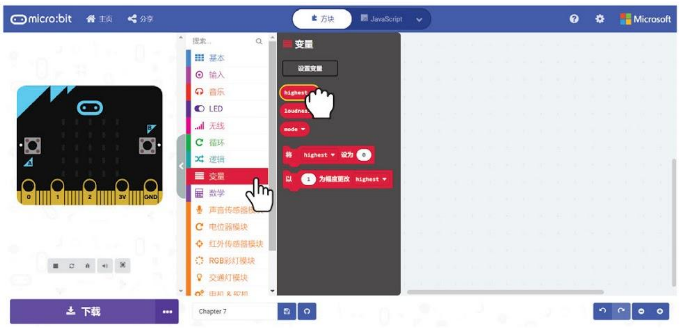

第16步 单击 [Led] 类别，然后选择 [绘制条形图，值为 __ 最高 __ ] 的条形图。单击 [变量] 类别，然后选择 [loudness] 块。将块对齐到 [绘制条形图，值为 __ 最高 __ ] 块并将其值更改为1023。



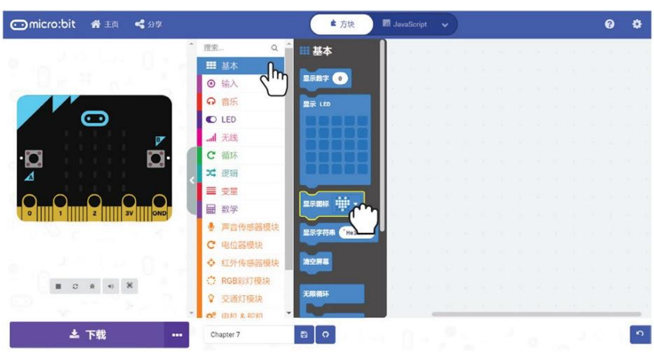


第17步 单击 [基本] 类别，然后选择[显示编号]块。将其对齐到 [如果–则–否则] 块的第二个插槽。从 [变量] 类别中获取 [highest] 块，并将其捕捉到 [显示数字] 块的值槽中。



第18步 单击 [基本] 类别，然后选择 [显示图标] 块。将其对齐到[如果–则–否则]块的最后一个插槽。



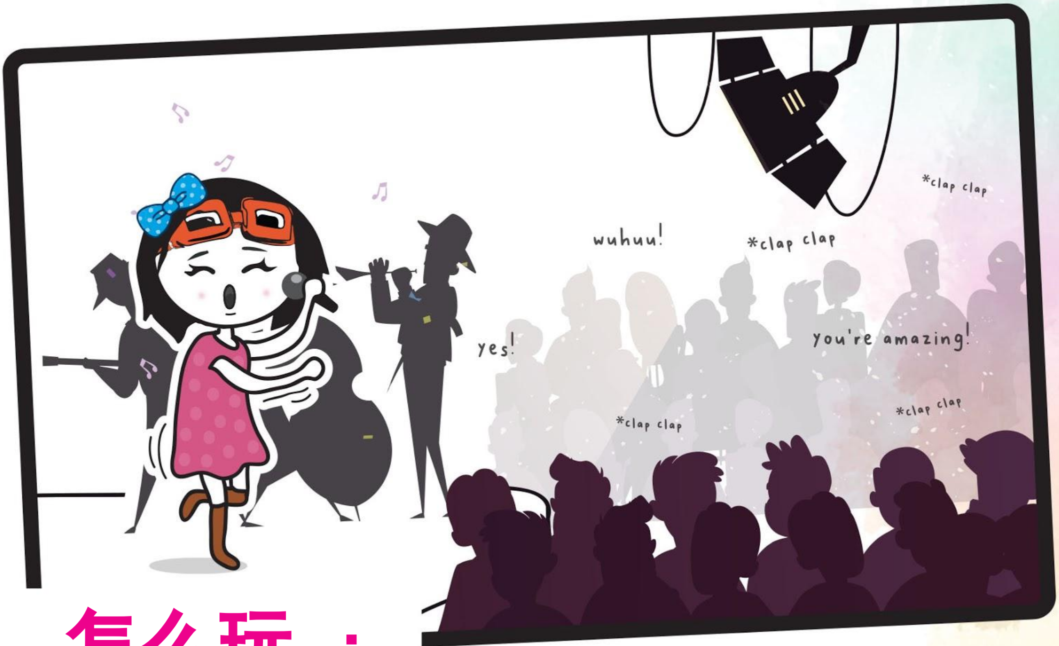
这是完整的代码：



第19步 将代码刷新到你的 EDU:BIT，你就可以为 “才华横溢时间” 节目准备一台掌声计。

# 让我们一起玩

听掌声响起！




## 怎么玩 ：

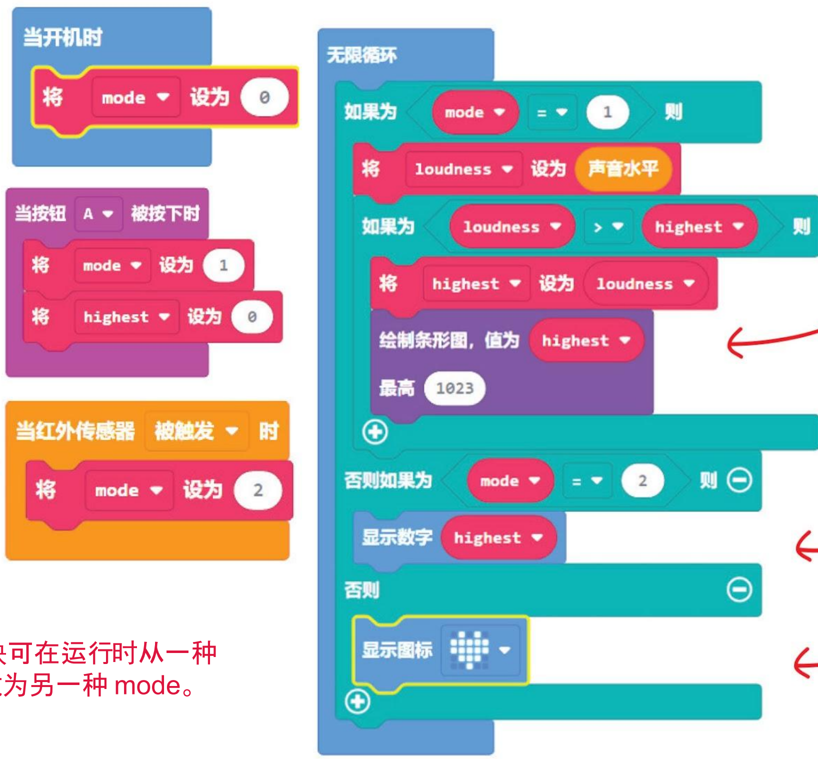
- 参赛者有时间单独或成对或成组参加一场简短的表演。你可以选择唱歌，跳舞或说一个笑话。
- 当每个人都准备就绪时，轮流展示你的才华。每次表演后，观众都会拍手回应 - 欣赏节目越多，拍手声音就越大。
- 掌声消退后，触发IR位以滚动乐谱（记录的最高声级）。
- 记得在下次演奏前按按钮A重设乐谱。
- 获胜者是得到最大声掌声的个人，对或团队。希望你玩得开心！

# 破解密码

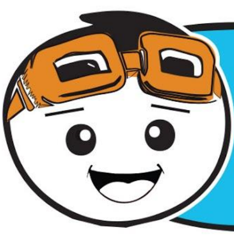
当我们在一个程序中有多个任务时，我们可以使用事件触发器从一个任务切换到另一个任务。为了使程序平稳运行，我们使用一个永远循环来不断检查当前模式，然后执行相应的代码块。

在启动时将 mode 设置为 0（待机模式）。

始终检查当前 mode 是什么，然后运行分配给该特定 mode 的任务 (代码块)。




事件触发块可在运行时从一种 mode 更改为另一种 mode。

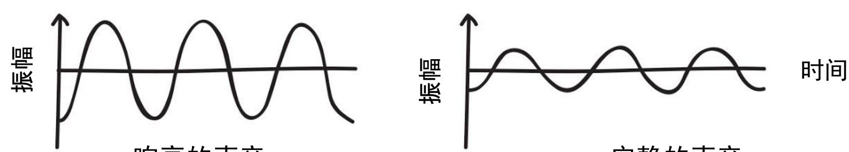


如果你要包含其他模式/任务，则始终可以在代码中添加更多事件触发块，例如 **[当振动]** 和 **[当声音水平 > __ 时]**，然后单击 **[如果-则-否则]** 块上的(+)按钮以添加其他条件。

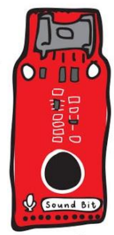
# 有趣的事实！

当物体振动时会产生声音，例如，敲击鼓时会产生声音。振动使周围的空气分子振动并产生声波。

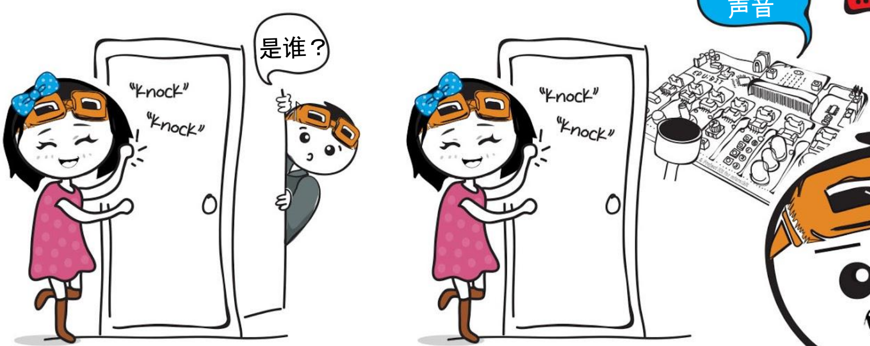
**声音传感器**是检测声波强度（响度）并将其转换为电信号的模块。

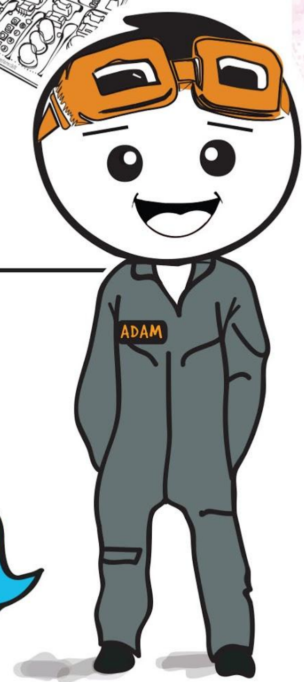



响亮的声音　　　　安静的声音




在某种程度上，声音传感器的工作原理类似于我们的耳朵，它将空气中的振动转换为电化学信号，大脑将其转换为我们所知的声音。




**以下是一些常见的用例**：

- 防盗报警噪声探测器
- 声控灯
- 音频婴儿监视器

你认为我们的声音传感器可以检测到太空中的任何噪音 吗？为什么或者为什么不？

# 编码挑战

程序 EDU:BIT 可用作教室的噪音监控器。点亮交通信号灯位上的 LED以指示噪声响度。

| 噪音响度 | 声音响度范围 | 点亮交通灯位 |
|---|---|---|
| 太吵了，请安静。 | （ ） 至 1023 | 红色LED |
| 有点吵。 请安静一点。 | （ ） 至 （ ） | 黄色LED |
| 可接受的噪音水平。 | 0 至 （ ） | 绿色LED |

**这里有一些提示给你。**
提示1 你将需要预先确定每个噪声水平的阈值。
提示2 为了获得更稳定的监控，请定期获取噪声水平测量值的平均值。

太容易了？ 试试这个挑战。修改你的代码，使阈值相对于电位计值。

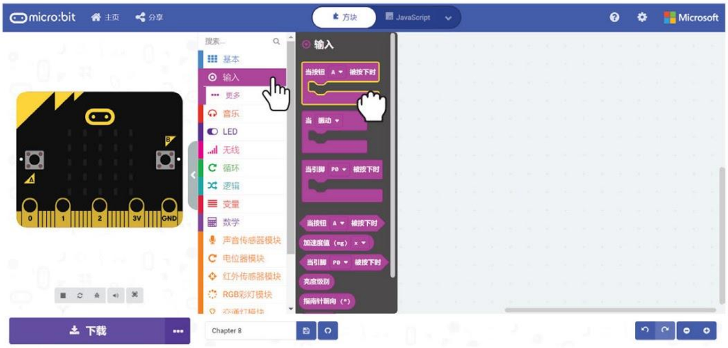
# 第8章

# 让我们一起旋转吧！

(DC Motor)

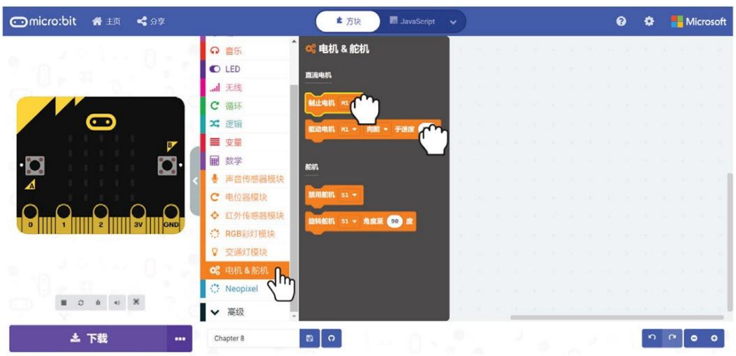
## 让我们编码！

第1步 在你的 MakeCode 编辑器中，开始一个新项目并添加 EDU:BIT 扩展名（你可以参考第40页）。单击 **[输入]** 类别，然后选择 **[当按钮 __ 被按下时]** 块。



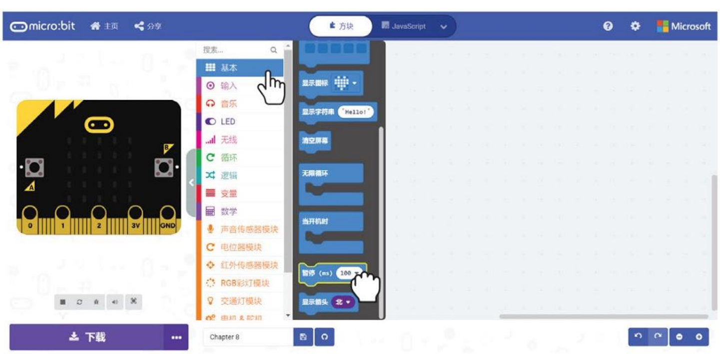


第2步 单击 **[电机& 舵机]** 类别，然后将 **[驱动电机 __ __ 于速度 __ ]** 块和 **[制止电机 __ ]** 块添加到您的代码中。将速度值更改为 **80**。

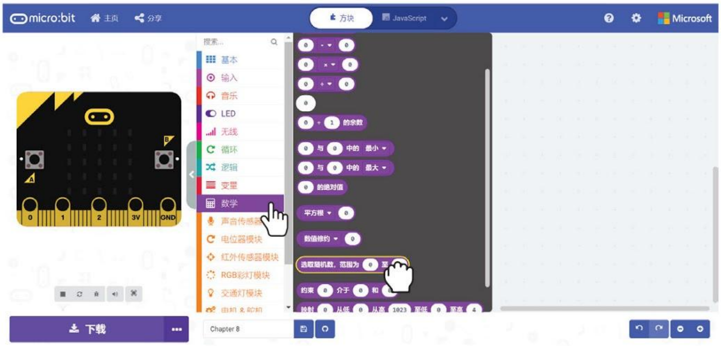




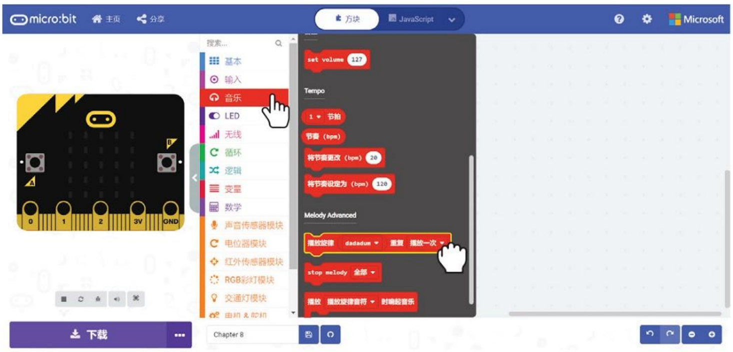
第3步 单击 **[基本]** 类别，然后选择 **[暂停 (ms) __ ]** 块。在 **[驱动电机 __ __ 于速度 __ ]** 块和 **[制止电机 __ ]** 块之间捕捉该块。




第4步 单击 [数学] 类别，然后选择 [选取随机数，范围为 __ 至 __ ] 块。将块放置在 [暂停 (ms) __ ] 块中，并将值分别更改为 200 和 1000。



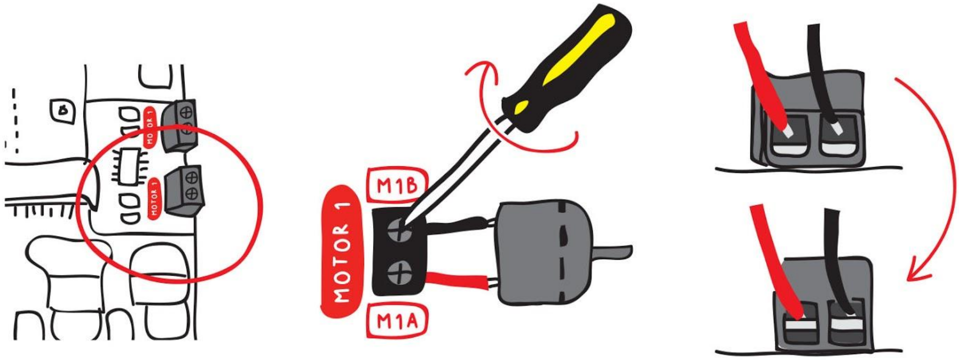

第5步 单击 [音乐] 类别，然后选择 [播放旋律 __ 重复 __ ] 块。将旋律更改为 “ba ding”（或选择所需的任何旋律）。

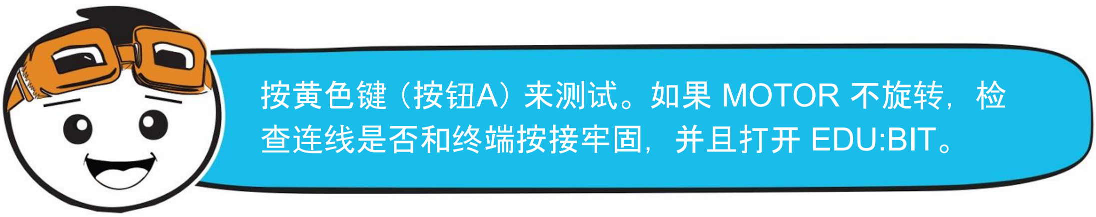



第6步 将完成的代码刷新到你的 EDU:BIT。

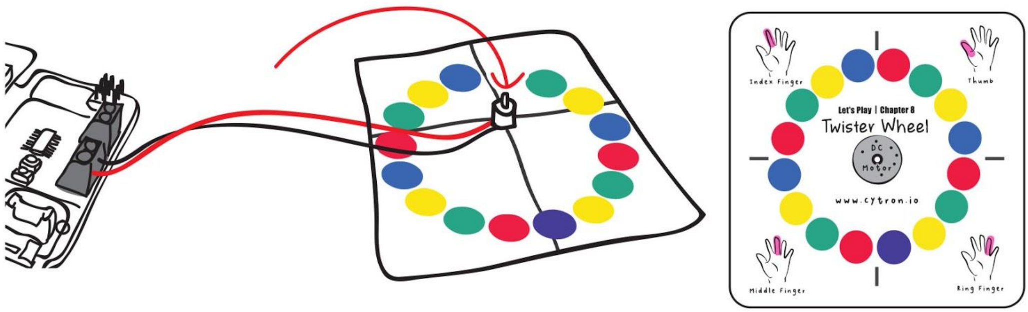

我们可以将此代码用于需要随机微调器的任何游戏。基本上，电动机在被触发时开始旋转，然后在随机周期后停止。

第7步 将直流电动机连接到 MOTOR 1 端子 -(i) 插入裸露的电线，然后(ii)使用提供的螺丝起子拧紧螺钉以固定连接并将其固定到位。

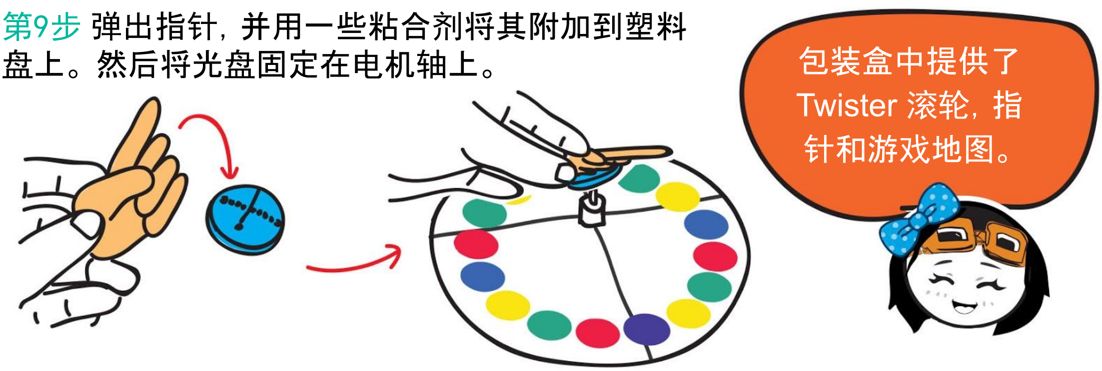



第8步 使用额外的粘合剂，例如双面胶带或热胶，将 DC 电动机连接到 Twister 轮的中心（如图所示）。




第9步 弹出指针，并用一些粘合剂将其附加到塑料盘上。然后将光盘固定在电机轴上。



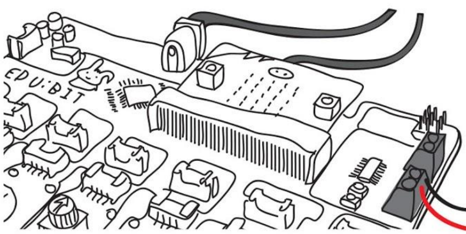
# 让我们一起玩

## 让我们一起旋转吧！

### 设置游戏


- 将游戏地图布置在桌子上。 两个玩家面对面坐着。 如果还有其他玩家，则他们可以占据剩余的边数（最多4位玩家）。

- 裁判坐在附近，手握EDU:BIT 扭轮。




### 怎么玩 ：

在此游戏中，玩家轮流按照裁判的指示将手指放在游戏地图上的彩色圆圈上。

裁判的角色是按下黄色按钮（或按钮A）旋转指针，然后调出手指和指针所指向的颜色，例如：“食指；红色”。

轮到你时，你必须听取裁判的指示，然后将指示的手指放在颜色正确的圆圈上。 如果该手指已经指示在颜色的圆上，则必须尝试将其移至相同颜色的另一个圆上。

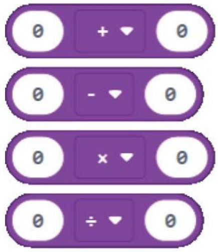
如果你未能成功完成移动，则你将退出游戏。

游戏中剩下的最后一个玩家是获胜者！

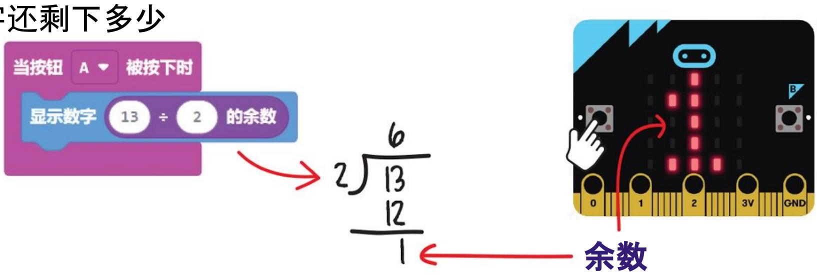
# 探索更多编码块

你可以使用 **[数学]** 类别中的块对变量执行算术运算。

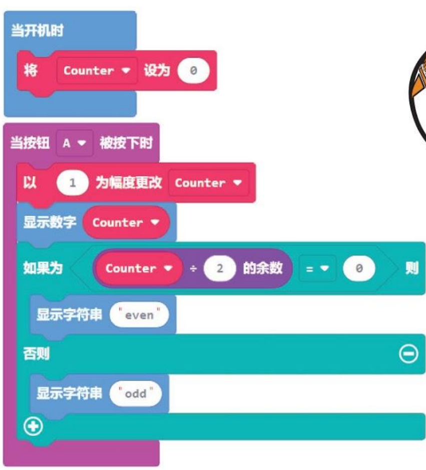
#1 使用以下块来加，减，乘或除。

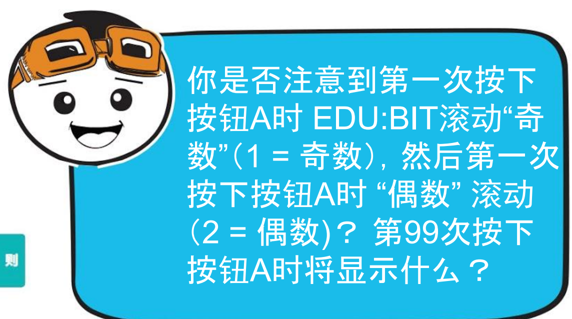



#2 你可以使用 **[ __ ÷ __ 的余数 ]** 块来查找如果一个数字除于另一个数字还剩下多少




#3 你还可以使用 **[ __ ÷ __ 的余数 ]** 块来确定数字是奇数还是偶数。只需除以2。如果余数为 “1”，则为“奇数”。如果余数为 “0”，则为“偶数”。我们试试吧！

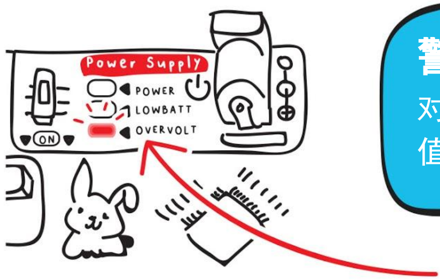


# 有趣的事实！

**直流电动机 (DC Motor)** 是将电能转换为机械能的旋转电气设备。

我们需要施加输入电压以使直流电动机旋转。我们可以通过调节输入电压来控制旋转速度。输入电压越高，电动机旋转得越快。EDU:BIT 中的直流电动机的推荐电压为 3.6V-6V。

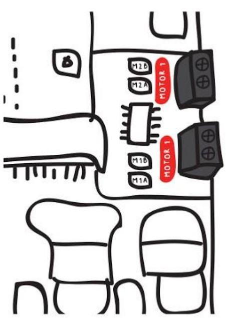



过压指示

**警告！**
对电动机使用高压（超出推荐值）会减少电动机的寿命。


你可以使用以下编程块轻松控制直流电动机的旋转方向和速度。


旋转方向




EDU:BIT 板上有两个直流电动机端子。选择正确的端子。

这是一个从 0 到255 的相对值。该值越高，电动机旋转越快。

你是否知道 EDU:BIT 具有内置的电动机测试电路？按下白色按钮（标记为 M1A，M1B，M2A 和 M2B）检查你的连接是否牢固，并且电动机工作正常。

# 编码挑战

| 将 EDU:BIT 编程为声控风扇，其速度由 Potentio Bit 控制。 | |
|---|---|
| 当开机时 | 显示心脏图标（或你选择的任何图标）。将变量 Mode 设置为 0。 |
| 当声音水平 > __ (阈值) 时 | 将变量 Mode 更改为1 |
| 无限循环 | 设置变量 ‘Speed’ 以将电位计值从低 0 高 1023 映射到低 0 高 255。<br><br>始终检查 Mode<br><br>• 如果 Mode 是偶数，则使用制动电动机 M1。<br><br>• 否则，如果 Mode 是奇数，则根据变量 ’Speed’ 的值（从电位计读数映射）运行电动机 M1。 |


**这里有一些提示给你...**

提示1：你将需要确定声 级触发器（例如，阈值）以激活和停止电动机。

提示2：你需要创建两个变量：Mode 和 Speed

提示3：将风扇叶片安装到电动机轴上，然后运行程序。如果在电机旋转时没有感觉到空气吹动，则需要在代码中更改旋转方向。

# 第9章

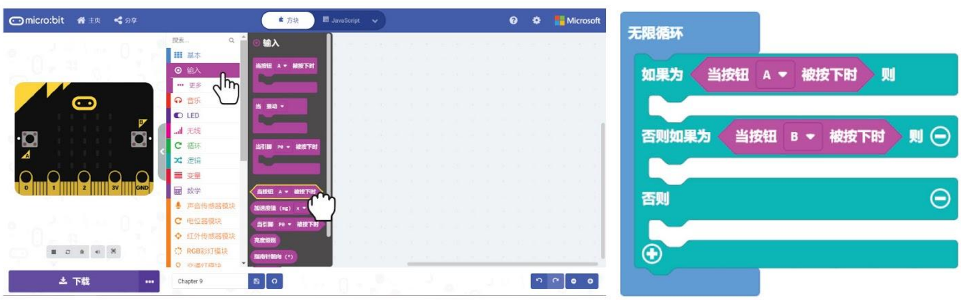
## 点球大战.. 得分!

(Servo Motor)

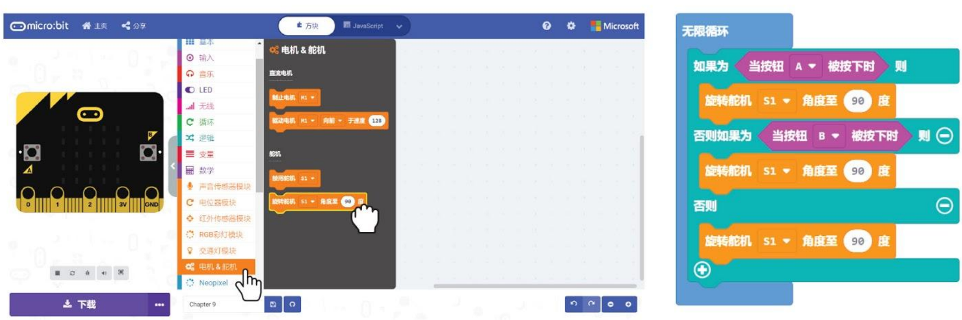
## 让我们编码！

第1步 在你的 MakeCode 编辑器中，创建一个新项目并添加 EDU:BIT 扩展名（你可以参考第40页）。单击［逻辑］类别，然后选择［如果–则–否则］块。将模块卡入［无限循环］插槽。单击加号图标以将另一个“否则”条件添加到块中。


第2步 单击［输入］类别，然后选择［当按钮 __ 被按下时］块。复制该块并将其捕捉到［如果–则–否则］块的条件槽。将第二个块更改为“**按钮B**”。

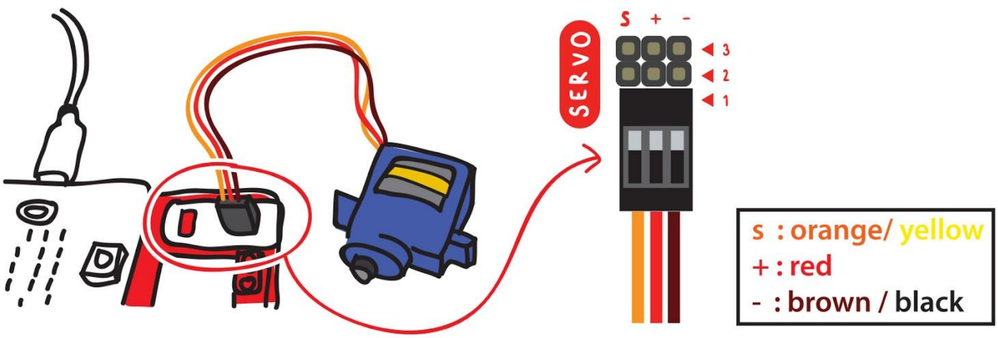



第3步 单击 [ 电机 & 舵机 ] 类别，然后选择 [ 旋转 能机 __ 角度至 __ 度 ] 块。复制该块并附加到［如果–则–否则］块的每个插槽。

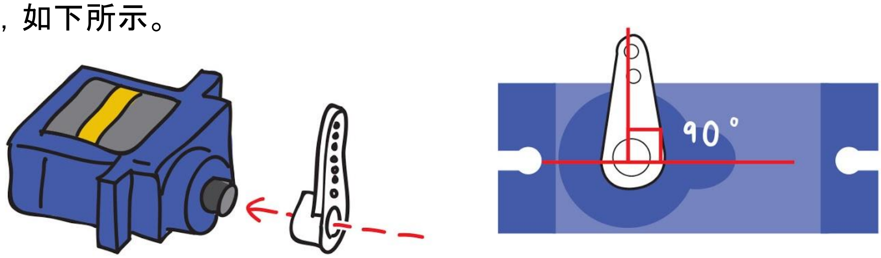


第4步 将第一和第二块的值分别更改为 **30** 和 **150**。将代码刷新到你的EDU:BIT。

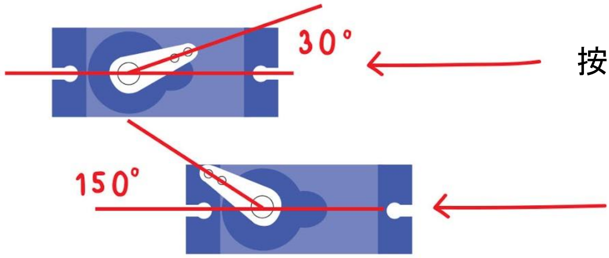

第5步 如图所示，将伺服电机电缆插入EDU:BIT上的 Servo Port 1。



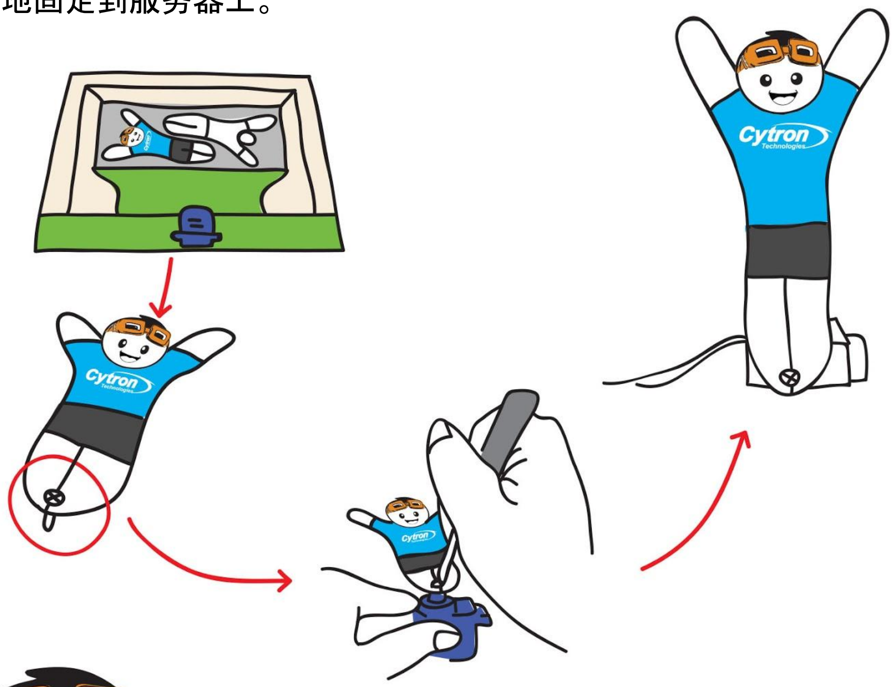

第6步 通 EDU:BIT 的电源，然后将伺服臂角以 90 度安装到电机伺服器的轴上，如下所示。



第7步 按按钮A，然后按按钮B进行测试。

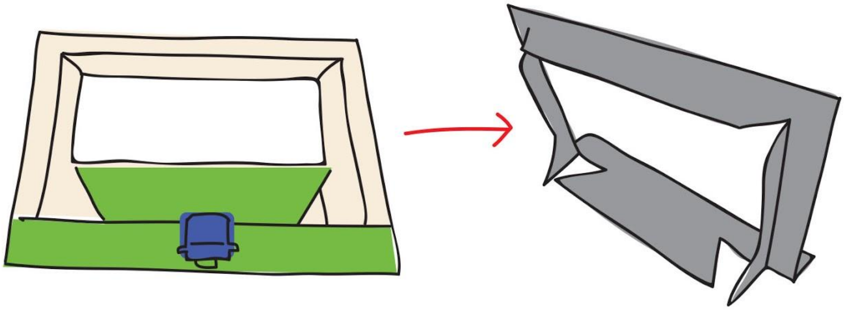



第8步 从提供的资源卡中弹出守门员。使用提供的螺丝刀和螺丝将守门员固定在伺服臂的喇叭上。使用额外的粘合剂，例如双面胶带或热胶，将门将牢固地固定到服务器上。

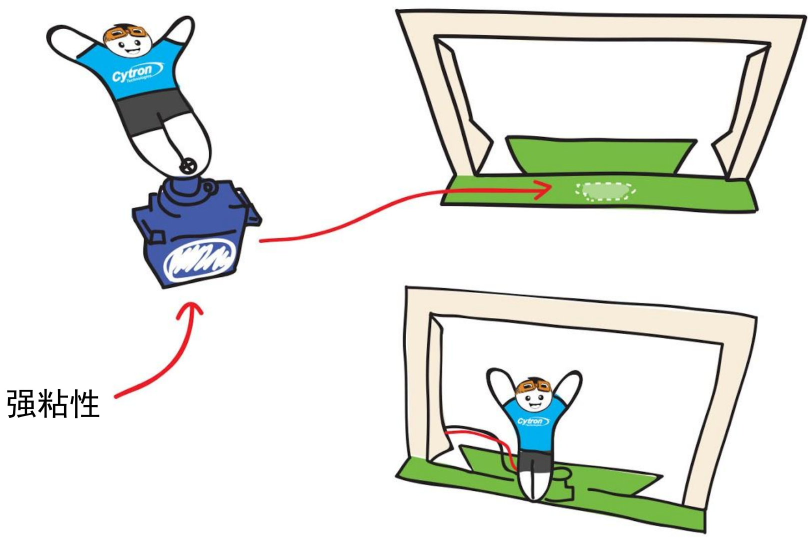


如果愿意，你可以使用提供的其他弹出窗口人物为门将设计自己的球衣。

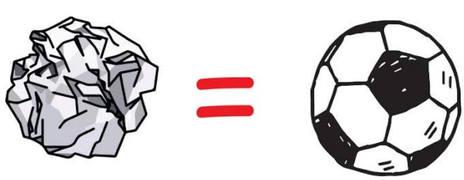
第9步 弹出目标球杆，如下所示进行设置。



第10步 如图所示，使用强力粘合剂（例如双面胶带或热胶）将伺服电机牢固地固定到位。

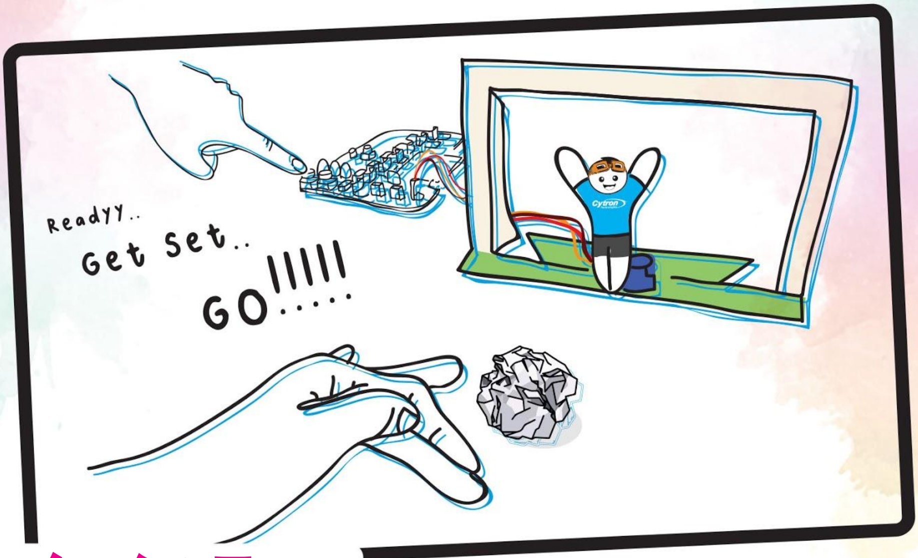



第11步 弄皱一小块纸就可以成为“足球”的替代品，我们全都准备好了进行点球大战的有趣游戏。 你准备好了吗？



# 让我们一起玩

## 点球大战.. 得分!




## 怎么玩 ：

设置球门柱并标记点球（距离球门约1米，为年轻球员调整距离）。

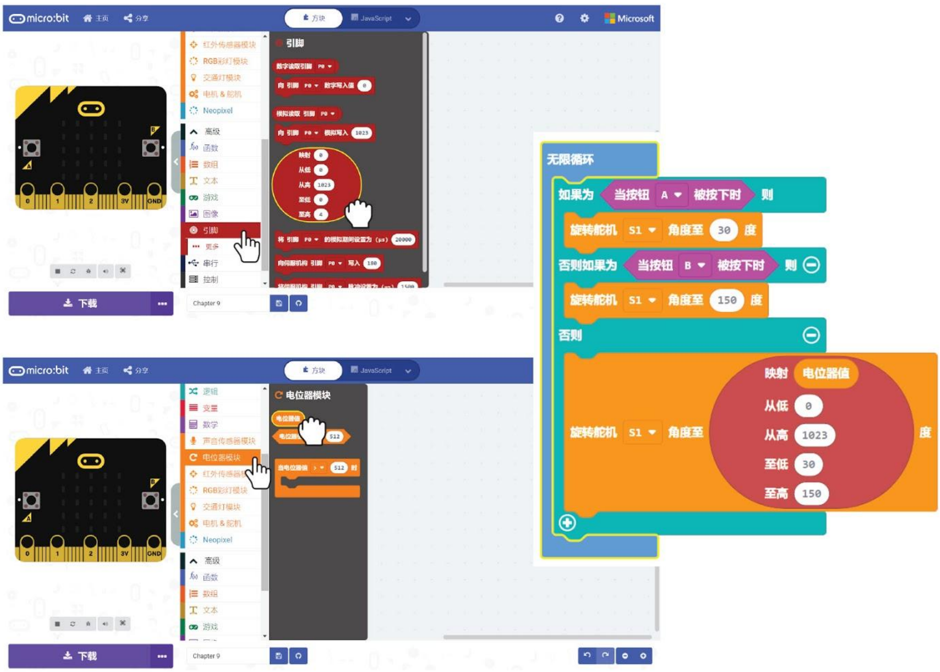
- 玩家轮流成为踢球者和守门员。
- 射手将球扑向球门。
- 守门员尝试通过按下黄色按钮（按钮A）向左移动或通过按下蓝色按钮(按钮B）向右移动来阻止球。
- 在一个回合中，每个玩家有5次机会。得分最高的球员是获胜者。

你知道吗？ 当比分在常规赛结束时被平局时，点球大战决定了足球比赛中的获胜球队，并且即使在加时之后仍然保持平局。在点球大战中，每支球队得到5球。球员从罚球朝着只有对方球队守门员看守的进球射击。获胜的球队将获得更多进球。

在前面的代码中，我们通过按按钮使守门员向左或向右摆动。我们可以使用电位位修改代码以控制守门员的位置。

第12步 单击 **[高级]** 类别，然后选择 **[引脚]** 类别。将 **[ 映射 __ 从低 __ 从高 __ 至低 __ 至高 __ ]** 块添加到你的代码中。

第13步 单击 **[电位器模块]** 类别，然后选择 **[电位器值]** 块。将块对齐到 **[ 映射 __ 从低 __ 从高 __ 至低 __ 至高 __ ]** 块，并将最后两个值分别更改为 **30** 和 **150**。



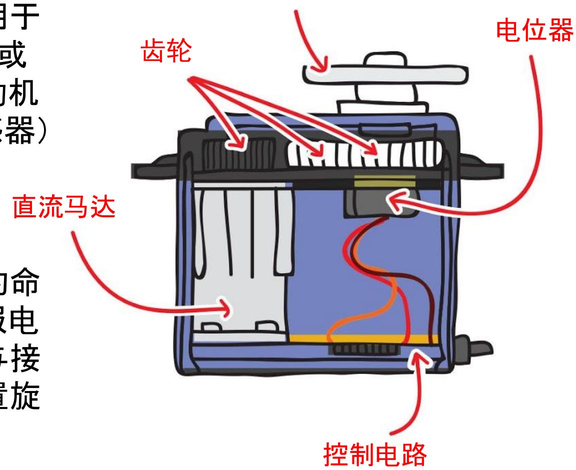

第14步 将代码刷新到你的 EDU:BIT。现在，你可以使用电位器控制守门员。希望你玩得开心!


如果你想自己练习踢罚球，可以通过使守门员不断向左和向右摆动，将代码修改为“练习模式”。试试看！

# 有趣的事实！

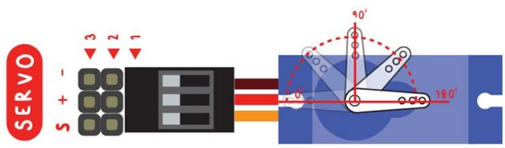
EDU:BIT 套件中的**伺服电机**也称为 RC（无线电控制）伺服器。它广泛用于遥控玩具车和小型机器人中以控制其运动。

服务器电机将三线系统用于电源（+），接地（-）和控制或信号。它通常由直流电动机，齿轮，电位计（位置传感器）和控制电路组成。




内置控制器将脉冲形式的命令转换为角度位置。伺服电机将不断朝着并保持在与接收到的脉冲相对应的位置旋转。

与连续旋转的直流电动机不同，我们可以将伺服电动机的旋转控制在 0 至 180 度范围内的所需角度。

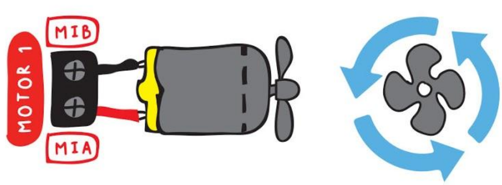



0-180 度旋转

**伺服电机**

**VS**

**直流电动机**




360 度旋转


youtu.be/okxooamdAP4

# 编码挑战

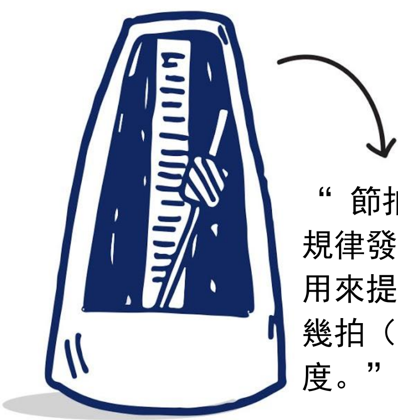
将 EDU:BIT 编程为节拍器。 通电后，将指针（固定在伺服电机喇叭上）向左摆动，然后以恒定速度反复向右摆动。 将速度设置为受潜在电位控制。 按下黄色按钮（按钮A）时，请显示当前速度（例如120 bpm）。


这里有一些提示给你。

提示1　你需要创建两个变量：Tempo 和 Delay。

提示2　典型的节拍器范围是 40 至 200 bpm。

提示3　速度为 60 bpm（或每分钟节拍）意味着指针在一分钟内从一端摆动到另一端总共摆动60次，例如每秒摆动一次。



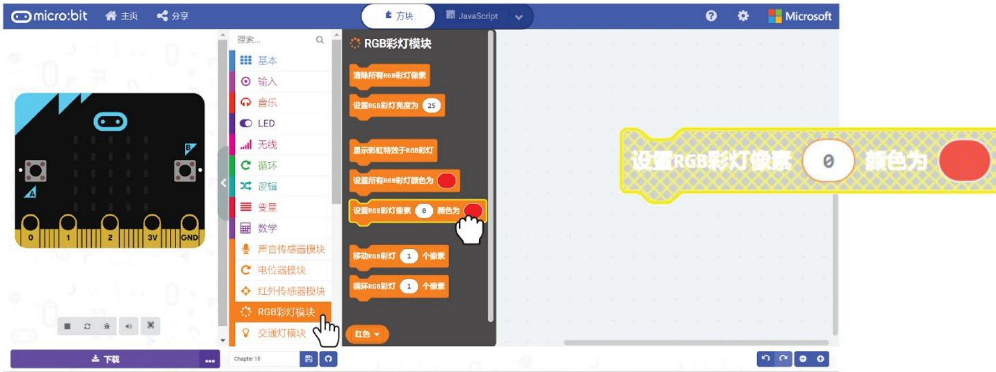
“ 節拍器（Metronome），又叫拍子機，是一種能規律發出聲音或視覺效果來顯示節拍的器材，通常用來提供一個穩定的節拍和速度；音樂上以每分鐘幾拍（beats per minute，bpm）來計算速度。” -- Wikipedia

第10章

# 主谋，你能破解密码吗？

(RGB Bit)

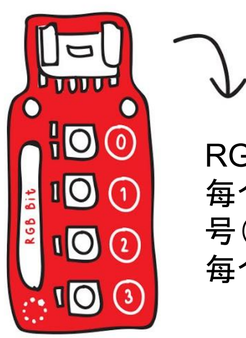
第1步 在你的 MakeCode 编辑器中，创建一个新项目并添加 EDU:BIT 扩展名。单击 **[RGB彩灯模块]** 类别，然后选择 **[设置RGB彩灯像素 __ 颜色为__ ]** 块。




第2步 在工作区中，右键单击 **[设置RGB彩灯像素 __ 颜色为__ ]** 块，然后单击“**复制**”。重复直到有四个 **[设置RGB彩灯像素 __ 颜色为__ ]** 块。



第3步 将这些块放置在 **[当开机时]** 块的插槽中。将第二，第三和第四块的 RGB 像素数从 0 更改为 **1**，**2** 和 **3**。


RGB Bit 上有4个 RGB LED，每个 LED 都分配有一个标识号(0-3)。使用此数字可分别对每个LED 进行编程。

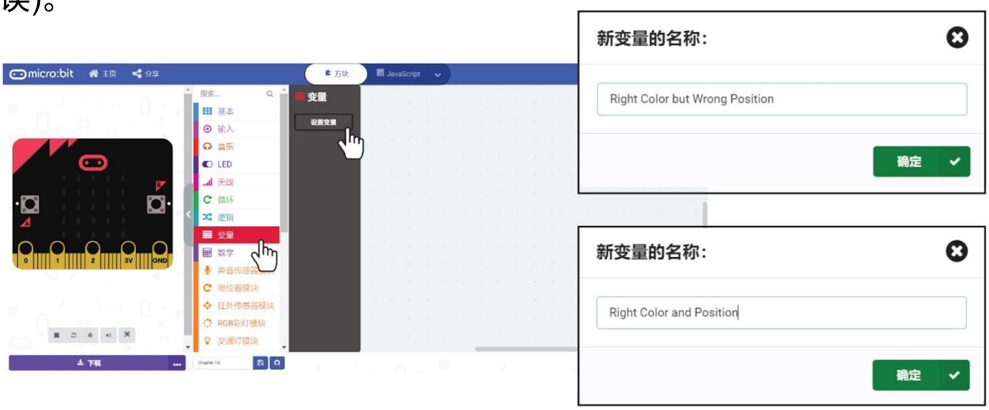
第4步 将代码刷新到你的 EDU:BIT。


当为板子加电时，RGB Bit 上的四个 LED 将根据设置的颜色点亮。

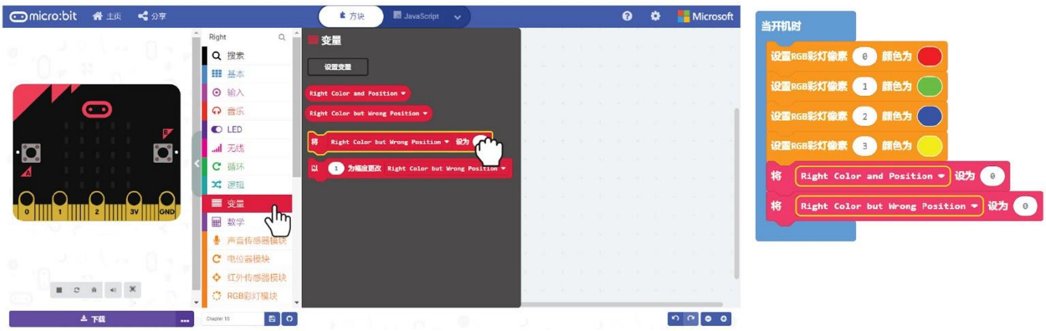
你可以通过单击块并从调色板中选择另一种可用颜色来轻松更改颜色。

试试看！


第5步 添加两个新变量并命名它们 - “**Right Color and Position**” (正确的颜色和位置) 和 “**Right Color but Wrong Position**” (正确的颜色但位置错误)。

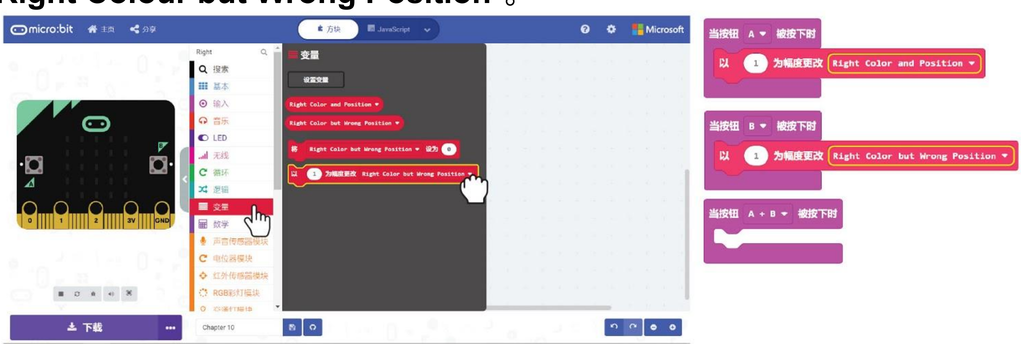


**第6步** 单击 **[变量]** 类别，然后选择 **[将 __ 设为 __]** 块。复制该块并将其捕捉到 **[当开机时]** 块中。将一个变量设置为 “**Right Color and Position**”，将另一个变量设置为 “**Right Colour but Wrong Position**”。

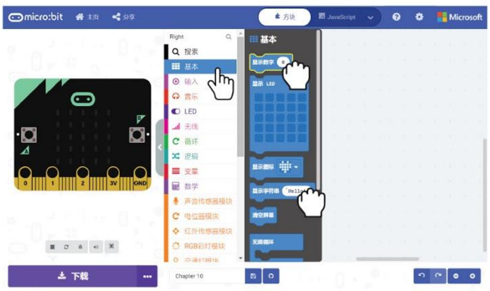



**第7步** 单击 **[输入]** 类别，然后选择 **[当按钮 __ 被按下时]** 块。复制块。将第二个块更改为**B**，将第三个块更改为**A + B**。


**第8步** 再次单击 **[变量]** 类别，然后选择 **[以 __ 为幅度更改 __ ]** 块。复制这些块并将其附加到 **[当按钮 A 被按下时]** 和 **[当按钮 B 被按下时]** 块。将一个变量设置为 “**Right Color and Position**”，将另一个变量设置为 “**Right Colour but Wrong Position**”。



**第9步** 添加［基本］类别中的［显示数字］和［显示字符串］块，以及［变量］类别中的[正确的颜色和位置]和[正确的颜色但位置错误]块。



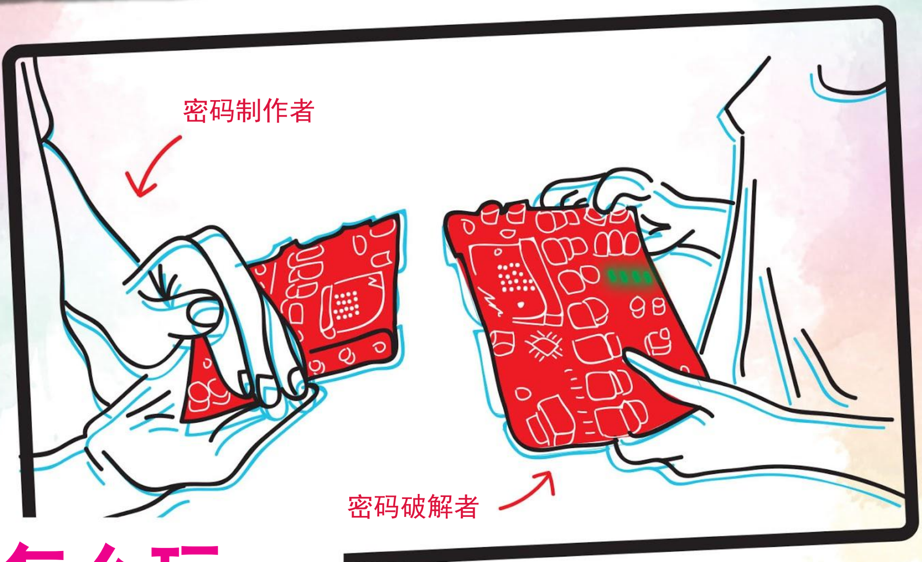
**第10步** 更改［显示字符串］块中的 “Hello!” 为 “**A =**” 和 “**B =**”。这是完整的代码。




**第11步** 将代码刷新到你的 EDU:BIT。你尚未准备好玩 “主谋” 游戏。

# 让我们一起玩

主谋，你能破解密码吗？




## 怎么玩 ：

玩家1 (密码制作者) 通过以随机顺序点亮四个LED来设置密码。开始时将颜色限制为红色和绿色。隐藏您的代码，以便其他玩家看不到您的设置。

玩家2（密码破解者）尝试猜测密码。他相应地在EDU：BIT上设置了RGB LED，并将其显示给密码制作者。

密码制作者检查并按密码破解者的EDU:BIT上的黄色按钮(A）和/或 蓝色按钮(B)，以指示密码破解者有多少个LED是 “Right Color and Position” 或“Right Color but Wrong Position”。

然后，密码破解者可以同时按下两个按钮以读取结果。

重复步骤2和3，直到代码破解者正确猜出颜色顺序为止（每轮最多10次尝试）。

交换角色并进行另一轮比赛。

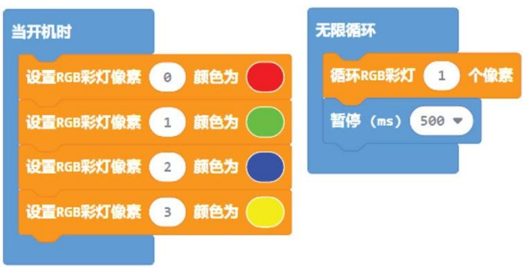
## 如何取胜

如果您能够成功破解代码（猜出正确的颜色顺序），您将获胜。如果您没有做到，那么胜利将归功于代码制作者。

# 探索更多编码块

#1 你可以通过将 **[设置RGB彩灯像素 __ 颜色为__ ]** 块替换为 **[显示彩虹特效于RGB彩灯]** 块，将 RGB 位上的LED设置为显示彩虹的颜色。


#2 你可以通过在 **[无限循环]** 块中放置 **[循环 RGB 彩灯 __ 个像素]** 块来创建流光效果。请记住添加 **[暂停（ms）__]** 块来减慢程序速度，以便你可以看到效果。这是一个示例代码：

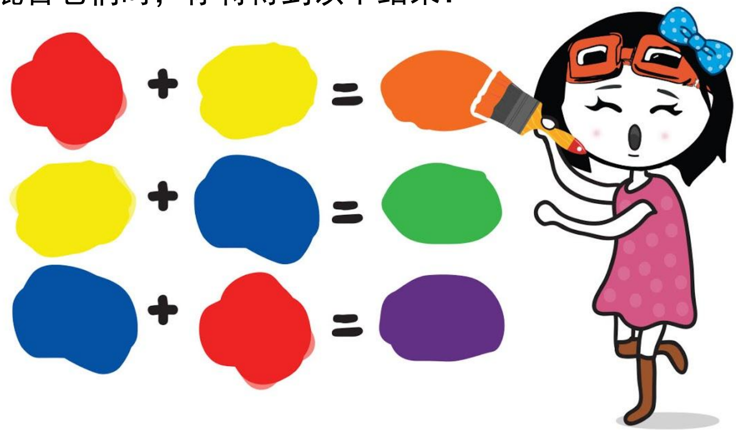



#3 你也可以使用 **[无限循环]** 块中的 **[移动RGB彩灯 __ 个像素]** 块将像素一一移位。你将需要添加 **[暂停（ms）__]** 块来减慢程序速度，以便可以看到效果，由此像素将被一个接一个地关闭。这是一个示例代码：

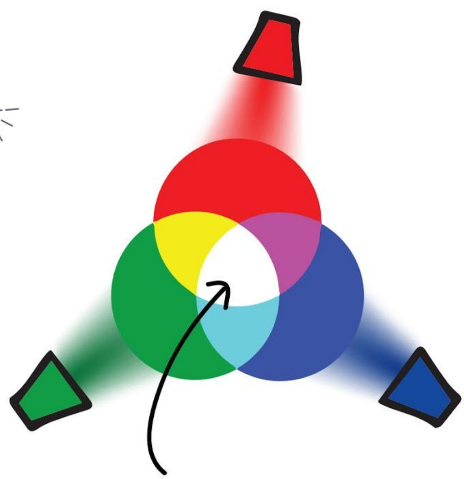

通过将设置从正值更改为负值，可以更改上述 #2 和 #3 的效果方向。

# 有趣的事实！

在画画课中，你可能会了解到3种原色是红色，黄色和蓝色。当你混合它们时，你将得到以下结果：



但是，对于使用灯光显示颜色的设备（例如你的电视和计算机屏幕），将使用 RGB 颜色模型。



在此模型中，三个光原色分别是 红色(R), 绿色(G) 和 蓝色(B)。 当结合在一起时，它们会发出白光！

# 编码挑战

将 EDU:BIT 编程为记忆游戏训练工具。

## 这个怎么运作？

- 首先，向左倾斜 EDU:BIT 以随机颜色模式点亮 RGB Bit 上的 LED 几秒钟。
- LED熄灭后，你必须观察然后说出颜色顺序。
- 要检查你的答案，请按蓝色按钮（按钮B）以点亮同一模块中的 RGB LED。
- 如果回答正确，请按黄色按钮（按钮A）更新并显示你的分数。如果回答不正确，请游戏结束。
- 你可以通过旋转 Potentio Bit 的旋钮来调整游戏的难度级别，以增加/减少 LED 持续点亮的持续时间。
- 得分最高的玩家获胜！

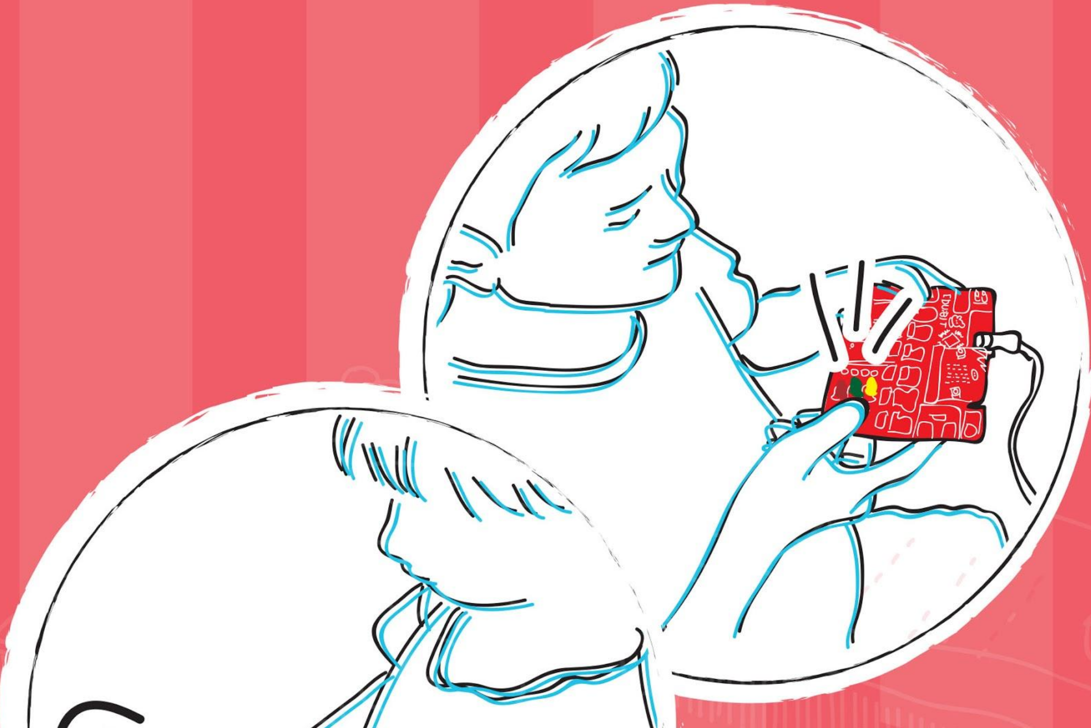
这里有一些提示给你：

提示 #1：你需要创建两个变量-得分和模式。
提示 #2：你需要为每个图案预先设置颜色顺序（每个RGB LED 的颜色）。 使用更多的颜色或预设更多的图案可以使游戏更具挑战性。 反之亦然，你可能希望限制年轻玩家的颜色/图案。

# 奖励章节

# Simon说

(无线电通讯)

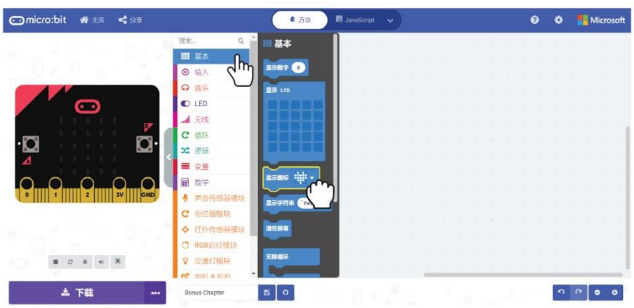


为了进行任何形式的通信，我们至少需要两个参与方-发送方和接收方。在这个游戏中，我们需要两个EDU:BIT 通过发送和接收无线电广播信号相互通信。

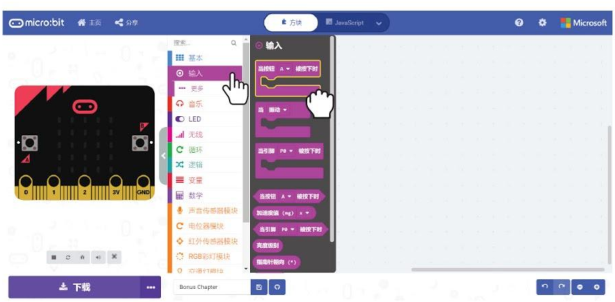
## 让我们编码！

第1步 在你的 MakeCode 编辑器中，创建一个新项目并添加 EDU:BIT 扩展名。单击 **[无线]** 类别，然后选择 **[无线设置组 __]** 块。将块对齐到 **[当开机时]** 块。

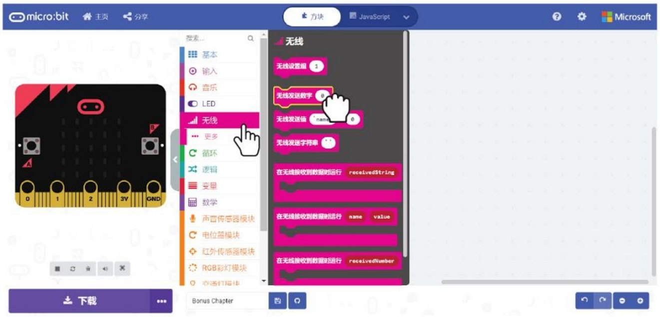

第2步 在你的 MakeCode 编辑器中，单击 **[基本]** 类别，然后在程序中添加 **[显示图标]** 块。



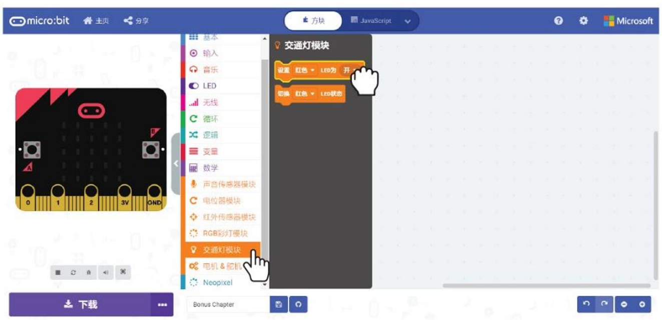

第3步 单击 **[输入]** 类别，然后选择 **[当按钮 __ 被按下时]** 块。



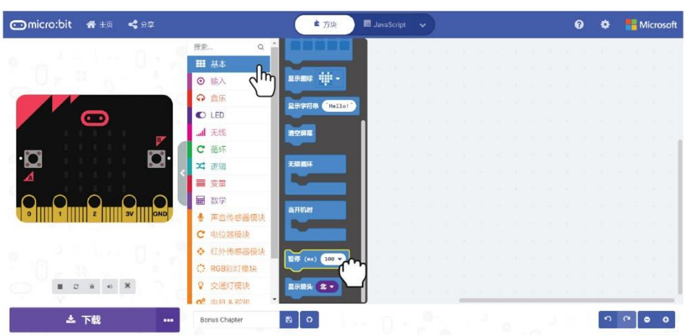
**第4步** 单击 **[无线]** 类别，然后选择 **[无线发送数字__]** 块。将该值更改为 **“1”**。




**第5步** 单击 **[交通灯模块]** 类别，然后选择 **[设置 __ LED为 __]** 块。复制它，并将两个块都捕捉到 **[当按钮 A 被按下时]** 块。将第二个块设置更改为 **“关”**。



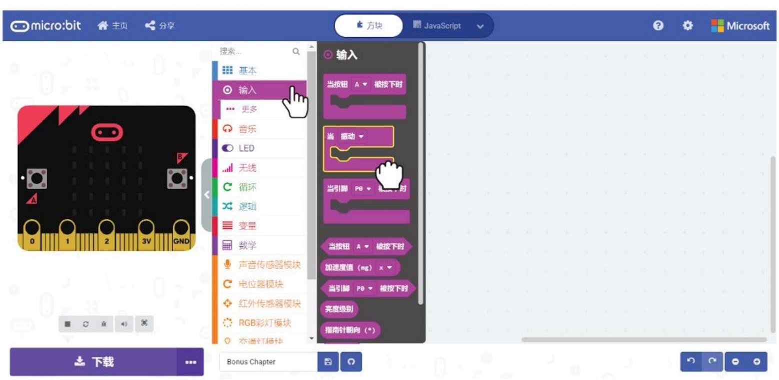

**第6步** 单击 **[基本]** 类别，然后选择 **[暂停（ms）__]** 。在 **[设置 __ LED为 __]** 块之间对齐该块，并将其值更改为 **500**。



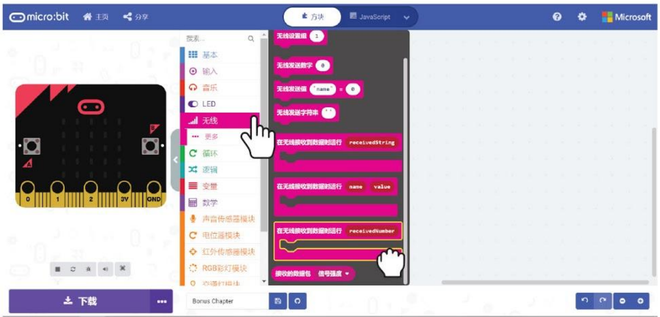
第7步 右键单击 **[当按钮 A 被按下时]** 块，然后选择“**复制**”。重复以获取三组相同的块。


第8步 将重复的代码块的按钮，数字值和 LED 颜色的设置更改为以下内容：


第9步 单击 **[输入]** 类别，然后选择 **[当 __]** 块。将设置更改为“**向左倾斜**”。单击 **[无线]** 类别，然后选择 **[无线发送数字__]** 块。将该值更改为 **4**
。




第10步 单击 **[无线]** 类别，然后选择 **[在无线接收到数据时 receivedNumber __ ]** 块。

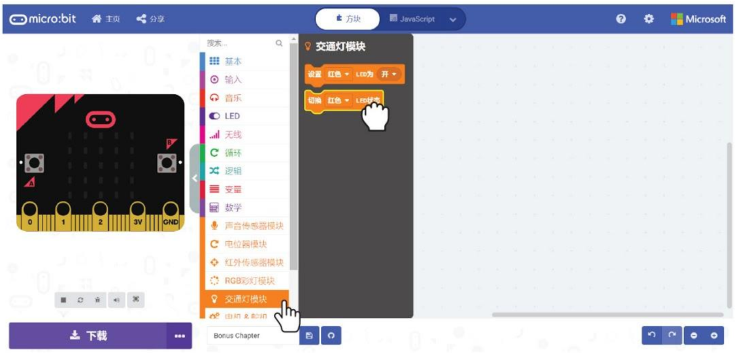



第11步 单击 **[逻辑]** 类别，然后选择 **[如果–则–否则]** 块。单击“ +”按钮添加三个 **[否则如果为]** 条件，然后单击 “-” 按钮删除 **[否则]** 条件。将 **[逻辑]:[ __ = __ ]** 比较块附加到每个“如果” 和 “否则如果为” 插槽。



第12步 单击并将 “**receivedNumber**” 变量拖放到比较块中，如图所示，并将其值分别更改为 **1**、**2**、**3** 和 **4**。

第13步 单击 [交通灯模块] 类别，然后选择 [设置 __ LED为 __] 块。复制该块并将其卡入前三个插槽。如图所示，更改颜色和开/关设置。





第14步 单击 [基本] 类别，然后选择 [暂停（ms）__] 。复制并对齐 [设置 __ LED为 __] 块之间的块，并将值更改为 500。将 [基本]:[显示图标] 块添加到最后的 “否则如果为” 插槽，并将图标更改为“ 悲伤的脸”。


第15步 单击［音乐］类别，然后添加［播放旋律 __ 重复 __ ］以完成你的代码。将旋律更改为“ **wawawawaa**”（或你选择的任何旋律）。

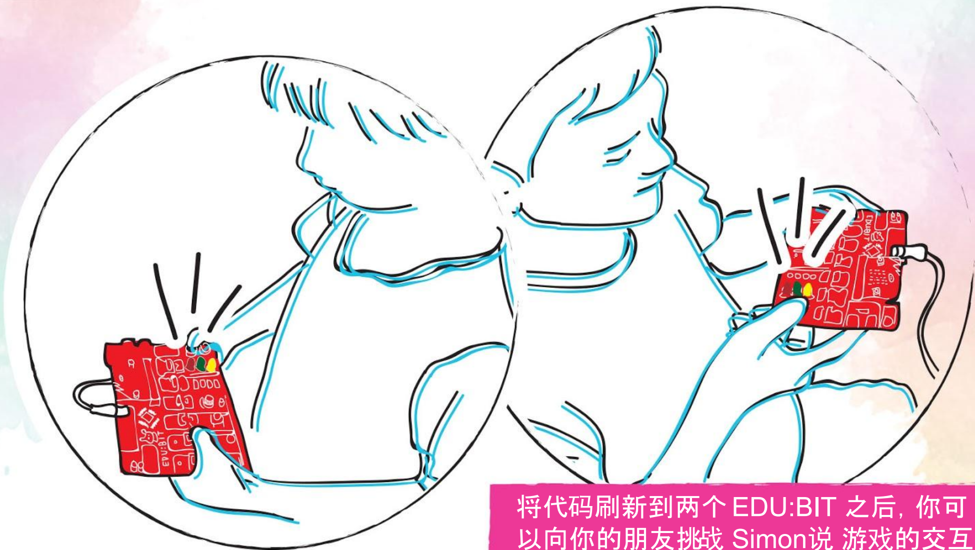
这是完整的代码


第16步 将完成的代码刷新到你的 EDU:BIT 以及你的朋友的代码中。

当两个 EDU:BIT 都打开时，你可以发送无线电信号来点亮朋友 EDU:BIT 上的 LED；反之亦然，你的朋友也可以按他的 EDU:BIT 上的按钮来点亮板上的 LED。

# 让我们一起玩

Simon 说



将代码刷新到两个 EDU:BIT 之后，你可以向你的朋友挑战 Simon说 游戏的交互式版本。

## 怎么玩 ：

两位玩家轮流成为 “Simon”。轮到你时，按按钮可以点亮红色，黄色和绿色的 LED。

- 要开始游戏，玩家1点亮交通灯位上的一个 LED。
- 播放器2观察并点亮相同的 LED，然后点亮另一个 LED。
- 游戏继续进行，每个玩家轮流重复最新的序列，然后点亮另一个 LED 以添加到序列中。
- 如果任何玩家点亮错误的 LED (或错误的顺序)，那么对手将倾斜其 EDU:BIT 结束游戏。
- 然后，失败者将需要重置其 EDU:BIT 才能开始新游戏。

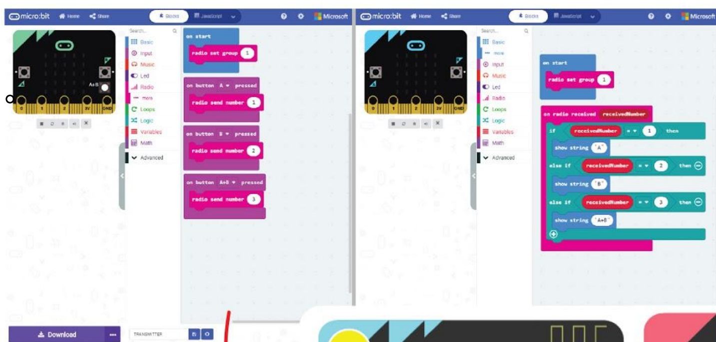
游戏一开始会很容易，但是每回合之后它会 变得越来越长，越来越复 杂。为了获胜，你需要仔细观察序列。这是一个有趣的游戏，可以训练你的记忆力。

# 探索更多编码块

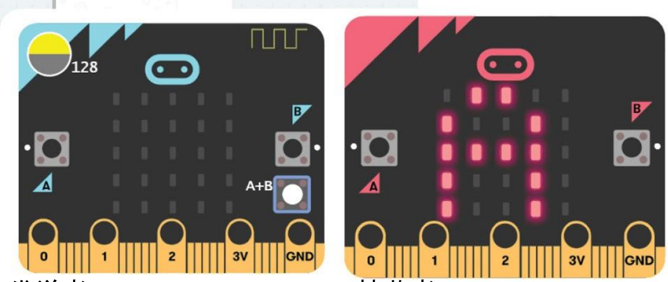
#1除了发送号码，你还可以使用 **[无线发送字符串 “ __”]** 块发送文本消息。你将需要使用 **[在无线接收到数据时运行 receivedString]** 块来接收字符串广播信号。字符串的最大长度为19个字符。

#2 如果需要一起发送和/或接收文本和数字，请使用 **[无线发送值 “ __ ” = __]** 块和 **[在无线接收到数据时运行 name value]**块。字符串的最大长度为8个字符。

如果你无权访问多个EDU:BIT，则仍可以测试无线电通信功能。只需访问 **makecode.com/multi** 来为 “发送者” 和 “接收者” 编写代码。你可以在相似器窗口中查看结果。



单击 micro:bit 板以全屏打开

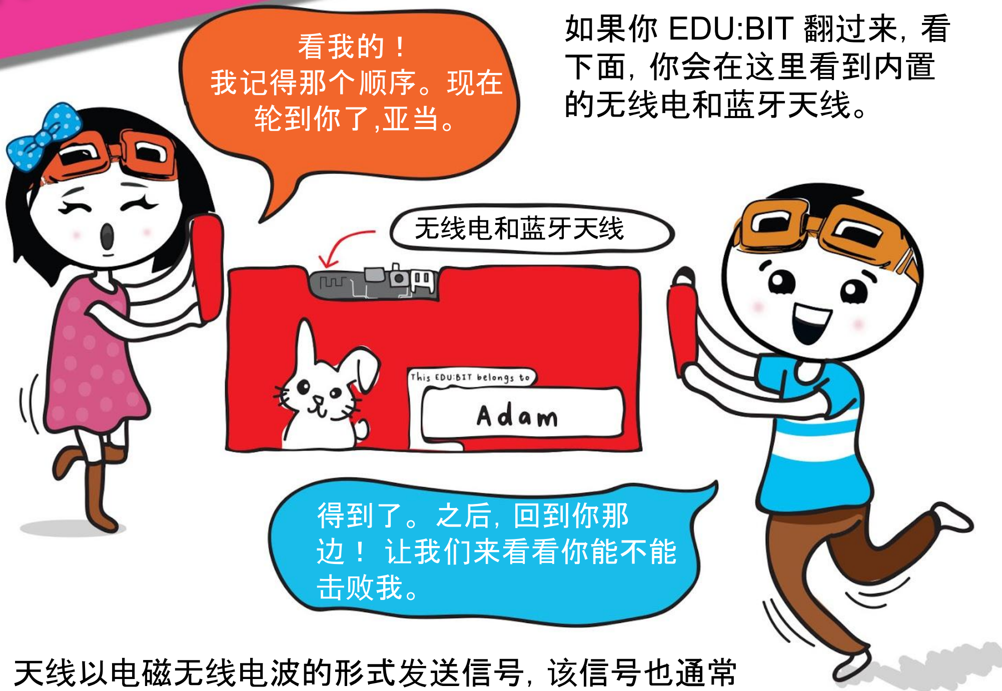



发送者　　接收者

当你在 “发送者” micro:bit 上按下按钮A时，“接收者” micro:bit 将接收无线电信号并显示接收到的字符串，例如A。如果你按下按钮A + B，会发生什么？

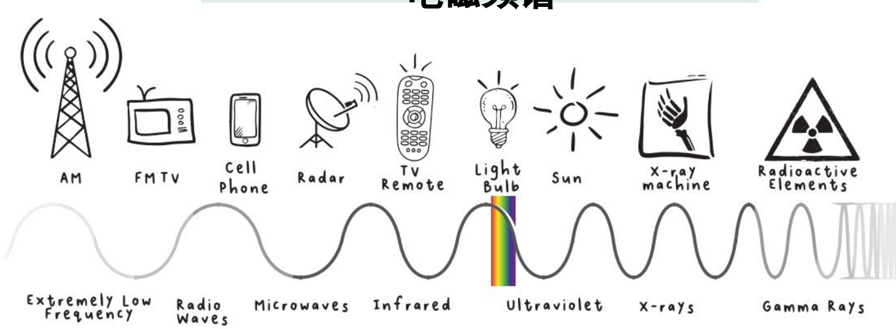
# 有趣的事实！

如果你 EDU:BIT 翻过来，看下面，你会在这里看到内置的无线电和蓝牙天线。




天线以电磁无线电波的形式发送信号，该信号也通常用于电视和无线电广播以及卫星传输。

## 电磁频谱




## 注意！

为了让你的 EDU:BIT 发送和接收来自其他 EDU:BIT的无线电广播信号，你需要将所有它们都设置为同一无线电组。

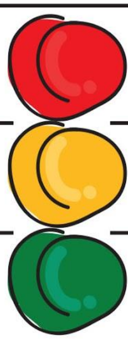
# 编码挑战

为你的班级建立反馈网络通信系统。

## 这个怎么运作？

将班级中的每个 EDU:BIT 都设置为相同的无线电组。

教师的 EDU:BIT 设置为在收到“字符串”信号时滚动文本，并在收到“数字”无线电信号时点亮交通灯位上的 LED，从而...

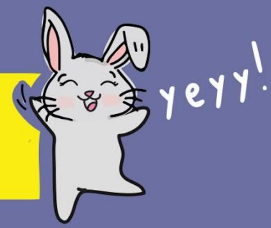
| 收到号码 | 点亮LED | 什么意思 |
|---|---|---|
| 1 | 红色  | A / 否/错误 |
| 2 | 黄色 | B / 也许/不确定 |
| 3 | 绿色 | C / 是的 / 对 |

将学生的 EDU:BIT设置为发送带有学生姓名的字符串，然后在触发时发送一个数字（1 或 2 或 3）以点亮教师 EDU:BIT上的 LED。

按按钮 A 发送号码 1。
按按钮 B 发送号码 2。
按按钮 A + B 发送号码 3。

这是给你的提示。为每个学生分配简短的昵称（只有两个或三个字符），以减少文本滚动时间。

# 我学会了...




- □ 程序EDU：BIT在LED矩阵上显示文本和动画。
- □ 下载，保存，发布和编辑 MakeCode .hex 文件。

- □ 使用输入块进行基于事件的编程
- □ 创建和使用变量

- □ 在Music Bit上使用压电蜂鸣器演奏简单的旋律。
- □ 创建和使用功能
- □ 阅读乐谱

- □ 程序 EDU:BIT 来控制交通信号灯上的 LED - 开启，关闭和切换。
- □ 向 MakeCode 编辑器添加扩展

- □ 编程 EDU:BIT 以使用IR位检测对象。
- □ 使用 while 循环
- □ 创建和使用数组

- □ 程序 EDU:BIT 从 Potentio Bit 读取模拟值
- □ 映射模拟输入读数
- □ 使用逻辑块进行条件编程

- □ 程序 EDU:BIT 可以用声音位检测噪声。
- □ 使用事件触发器在几种模式之间切换

- □ 程序EDU：BIT来控制直流电动机-旋转方向和速度控制
- □ 使用数学块来表现

- □ 程序EDU：BIT来控制伺服电机-角位置

- □ 程序EDU：BIT以不同的颜色和图案点亮RG Bit上的RGB LED

- □ 程序EDU：BIT发送和接收无线电信号

*如果你已熟练掌握该技能，请勾选该复选框；否则，请单击该复选框。否则返回相关章节进行修改。

# 恭喜你!!

你已经遍及所有章节，并学会了使用 MakeCode Editor 进行编码。 我们希望你玩得开心，玩了我们精选的流行儿童游戏。 并祝贺为你的教室开发了一些方便实用的应用程序。

到现在为止，你应该已经掌握了 micro:bit 以及 EDU:BIT 板随附的所有其他位的功能。 附...你知道你可以分解每个位吗？

继续并根据需要“分离”它们。 与主板分离后，你可以使用各种Bits构建新项目； 你需要使用提供的即插即用电缆连接它们。

现在是时候为新游戏和应用程序进行思考并进行头脑风暴了。 我们迫不及待想知道你将要提出什么惊人的项目。

请记住与我们分享你的创作。 通过电子邮件发送或在FB页面上给我们留言。 我们很乐意听取你的意见。

真诚的,

亚当和安娜


Bit..
Bit....