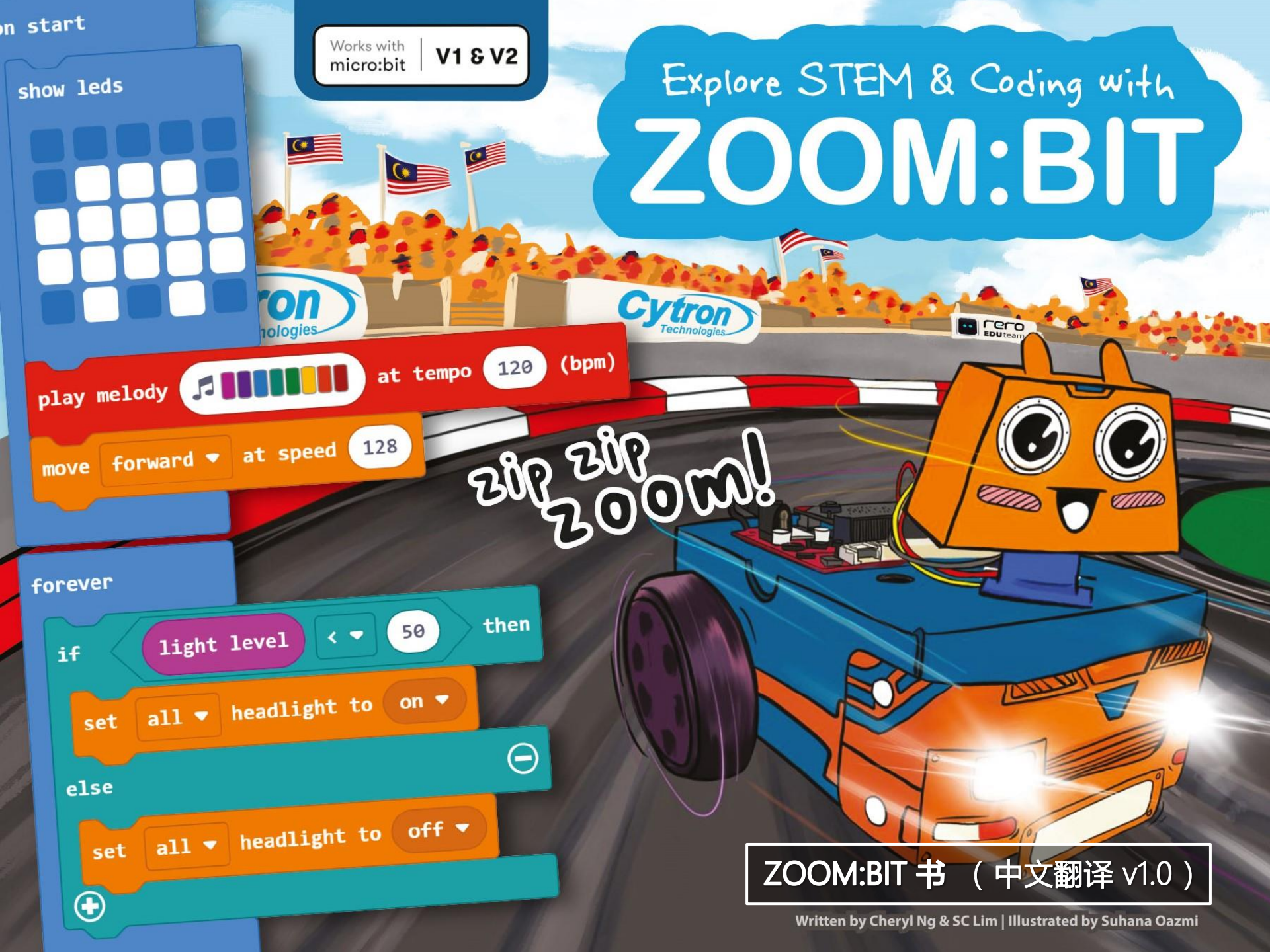

Works with micro:bit V1 & V2
Explore STEM & Coding with
ZOOM:BIT
on start
show leds
play melody at tempo 120 (bpm)
move forward at speed 128
forever
if light level < 50 then
set all headlight to on
else
set all headlight to off
Cytron Technologies
zip zip zoom!
ZOOM:BIT 书 （ 中文翻译 v1.0 ）
Written by Cheryl Ng & SC Lim | Illustrated by Suhana Oazmi

Cytron rero EDUteam 欢迎您！

亲爱的 Jr. Maker，

我们是亚当和安娜。欢迎加入我们创客·自造者 (Maker) 的行列。

在接下来的内容当中，我们将带领你一起建造 ZOOM:BIT，很快地你就会拥有自己的智能小车。你将会与我们一起学习编程与训练你的 ZOOM:BIT，并且用它来表演一些令人惊叹的才艺。它肯定能让你与朋友们玩得不亦乐乎。

https://t.me/zoombit_support

如果你面对任何问题，你可以浏览 Telegram 向我们求助，t.me/zoombit_support。
所以你准备好了吗？让我们开始吧！

亚当 和 安娜 上

使用 ZOOM:BIT 一起探索 STEM 与编程

作者
Cheryl Ng 与 SC Lim

绘图
Suhana Oazmi

中文翻译
Eng Zhi Qiang 与 Lim Zhe Yih

2021

出版/发行

Cytron Technologies

ISBN-978-967-19475-1-7

出版/发行
Cytron Technologies Sdn Bhd
No 1, Lorong Industri Impian 1,
Taman Industri Impian,
14000 Bukit Mertajam,
Pulau Pinang, Malaysia.
联络电话：+604-5480668

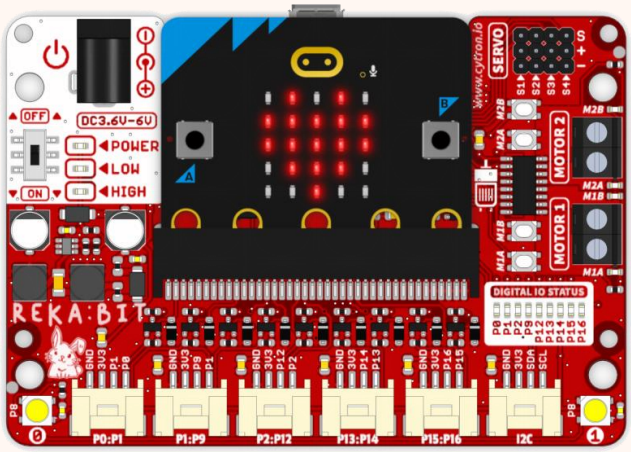
www.cytron.io/p-zoombit

马来西亚印刷

目录

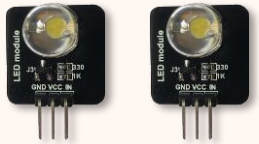
一起开箱吧！盒子里有些什么？

快速入门指南册

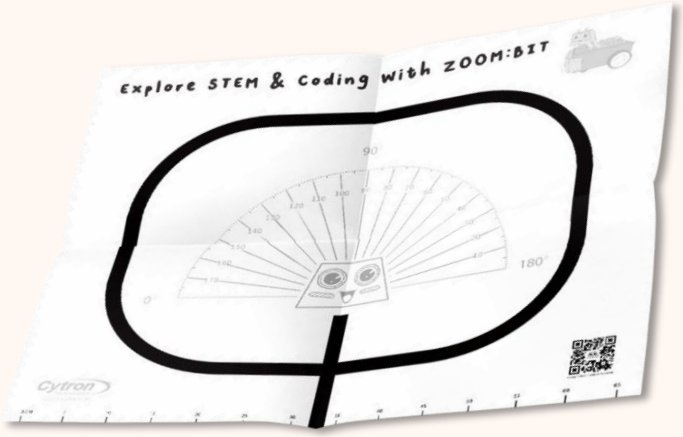


REKA:BIT（有或无 micro:bit）

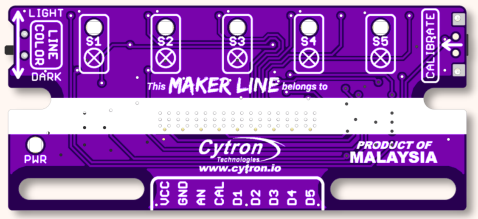
超声波传感器



LED 模块（车前灯）



ZOOM 跑道



Maker Line 巡线传感器

Grove 连接线

直流马达 和 轮胎 x2

螺丝刀

万向轮

螺栓，螺母 x4

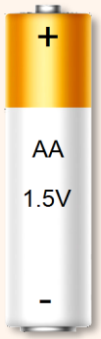
USB电源与数据线

伺服电机
Servo Motor

双面胶 x5

白铆钉 x4

黑铆钉 x8

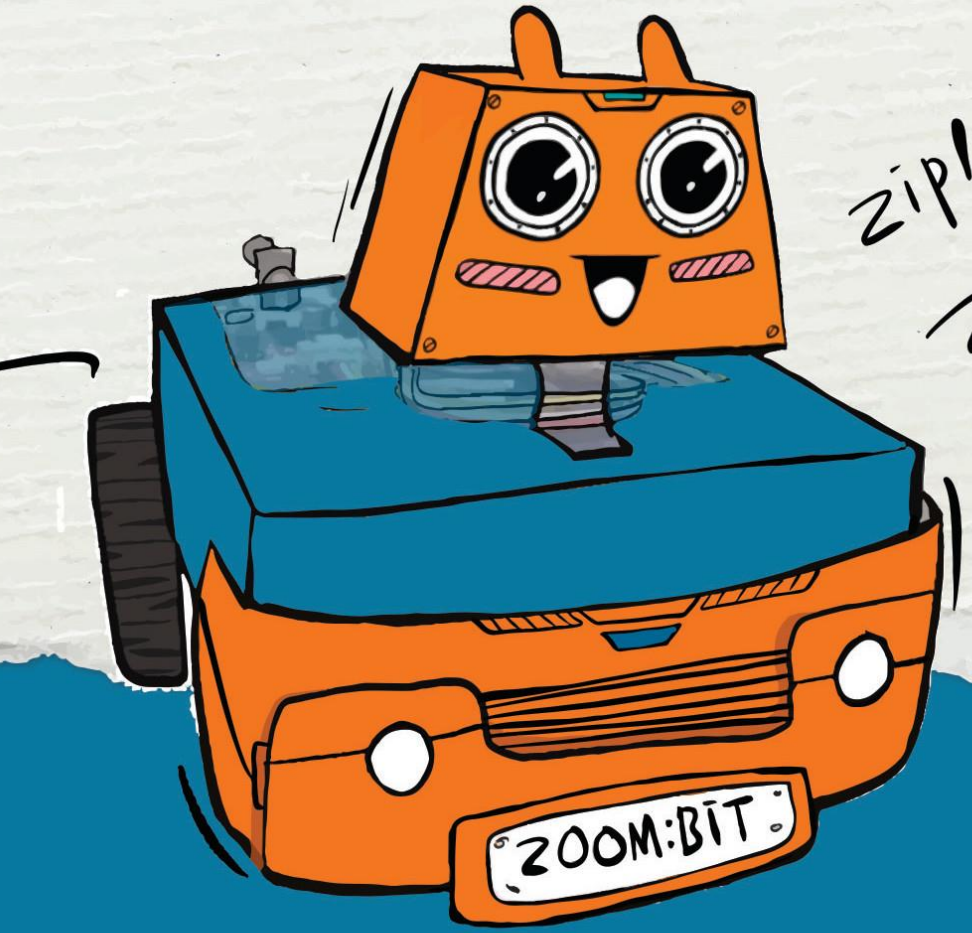


AA电池 x4

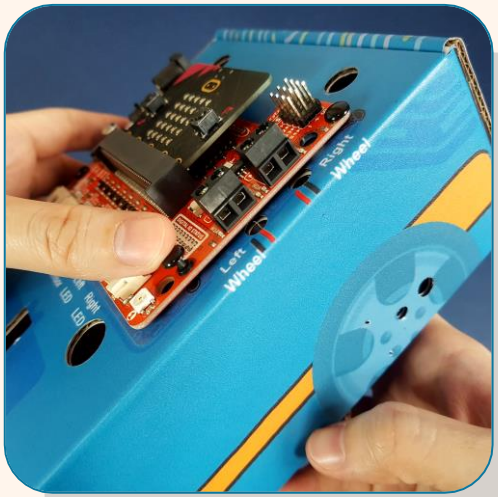
电池盒

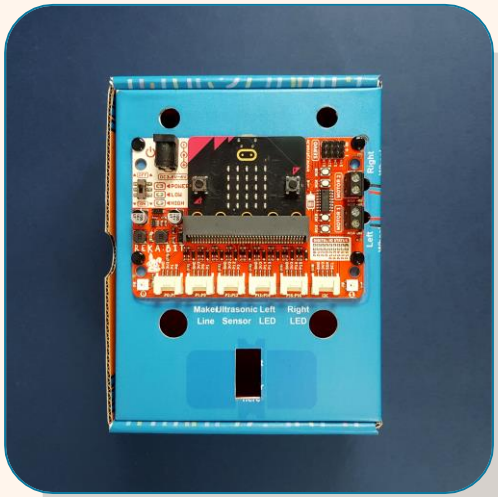
Let's Build!

来开始搭建!



https://link.cytron.io
/zoombit-resource-hub

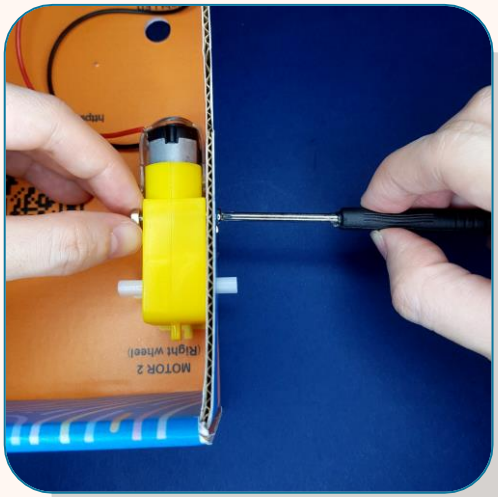




1. 把所有零件倒出来，把全部穿孔部分与盒子分离。

2. 把 REKA:BIT 放在盒子上面，用来对比和确认铆钉孔的位置。

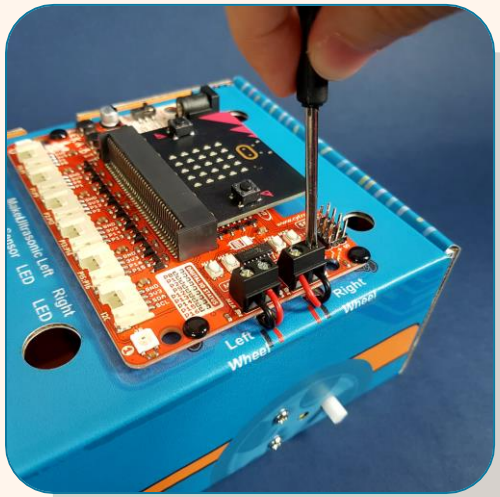
3. 把四个黑铆钉放入铆钉孔，然后用力按压来固定 REKA:BIT 的位置。

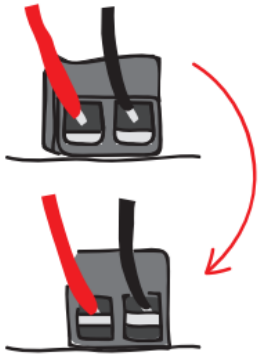


4 使用螺栓和螺母来固定**右**边的直流马达于标记着 MOTOR 2 的位置 (右轮) 。

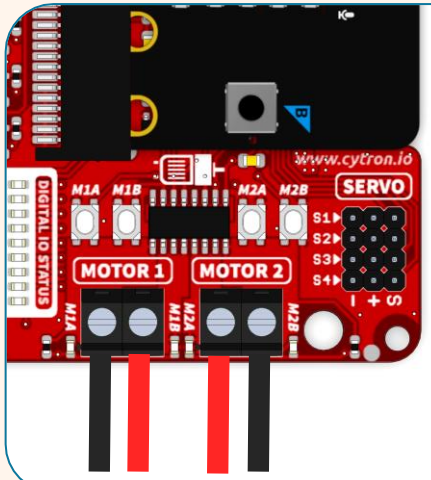
5 重复第四步，固定**左**边的直流马达于标记着 MOTOR 1 的位置 (左轮) 。

6 把电线穿过标记着 MOTOR 1 和 2 的洞口。





* 夹住裸露的导线
而不是绝缘体。



7

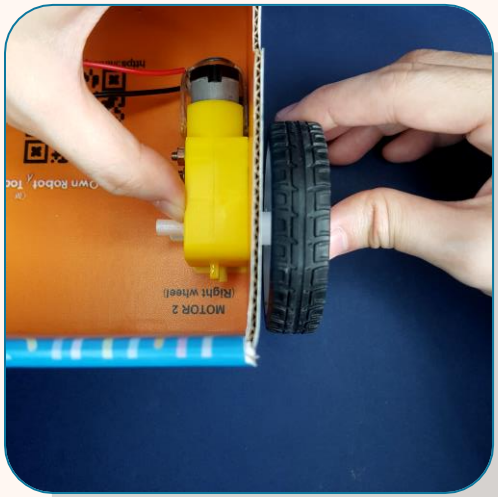
连接马达的电线于 REKA:BIT 上标记着 MOTOR 1 和 2 的终端：

（i）把裸露的电线心插入电线槽。

（ii）然后使用已提供的螺丝刀锁紧电线槽的螺丝，巩固连接的部分。

电线连接方式：

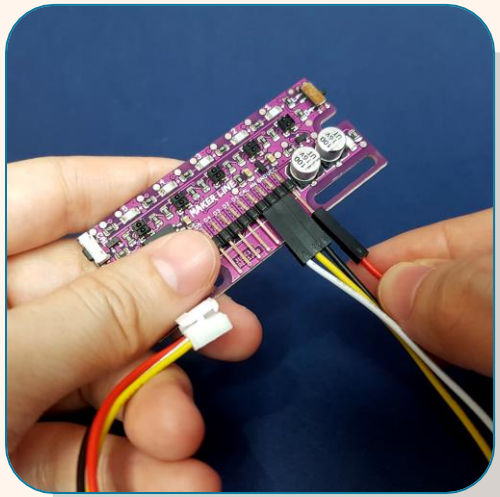
马达		马达终端
MOTOR 1	黑 (-)	M1A
	红 (+)	M1B
MOTOR 2	红 (+)	M2A
	黑 (-)	M2B

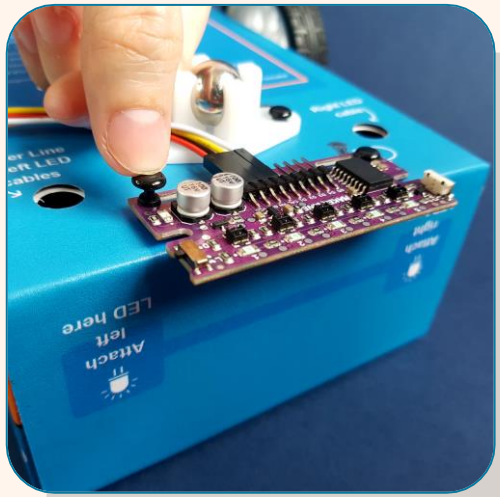


8 把左右边的轮胎安装到直流马达轴上。

9 把盒子倒转，然后安装万向轮到所显示的位置。

10 把两个黑铆钉放入铆钉孔，然后用力按压来固定万向轮的位置。





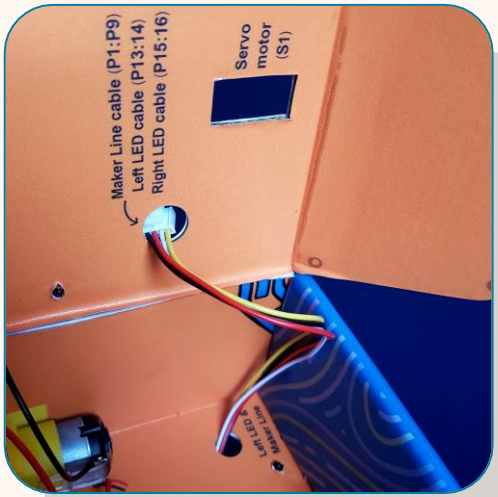
11 将一条 Grove to 4-pin female jumper 电线连接到 Maker Line 巡线传感器。

*如果你使用着 micro:bit 第一代 V1，请不需要连接白色的电线，保留着无连接状态即可。

12 把 Maker Line 巡线传感器放到指定的位置，然后使用两个黑铆钉固定它的位置。

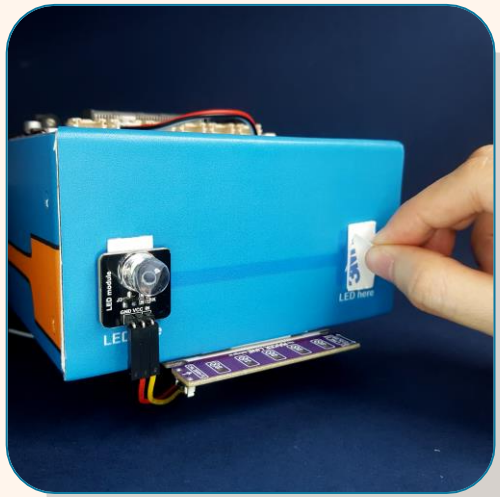
电线连接方式：

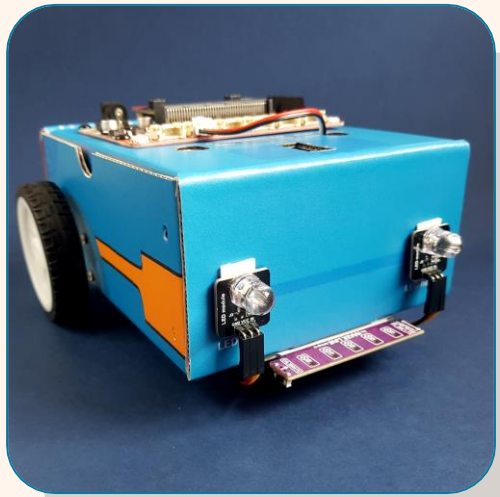
Grove 电线	Maker Line 巡线传感器
白	CAL (Calibrate)
黄	AN (Analog)
黑	GND (Ground)
红	VCC (Power input)



13 根据所显示的图案，把电线穿过盒子上的洞口。

14 把 Maker Line 巡线传感器的电线连接到 REKA:BIT 上 **P1:P9** 接口。





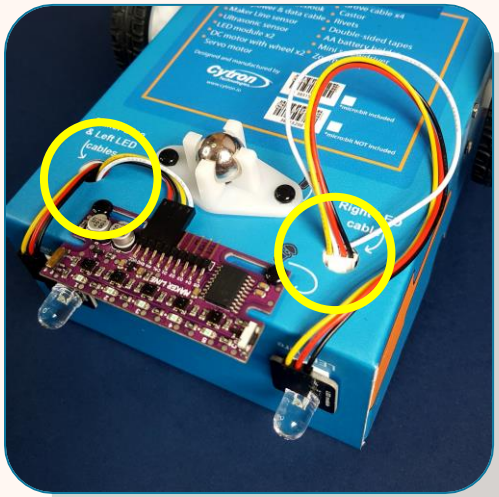
15 将一条 Grove to 4-pin female jumper 电线连接到 LED 模块上。

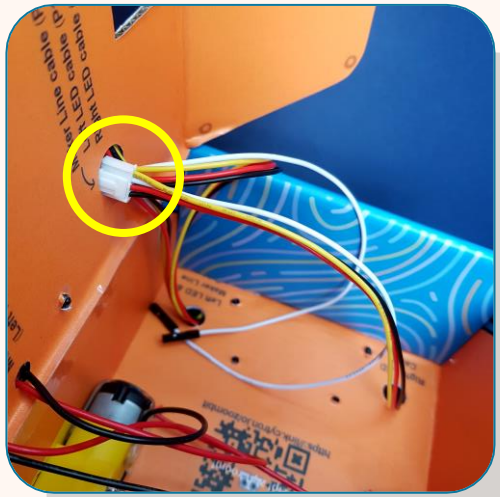
16 重复上一个的步骤，连接另一个 LED 模块。 **白色电线保留无连接状态**

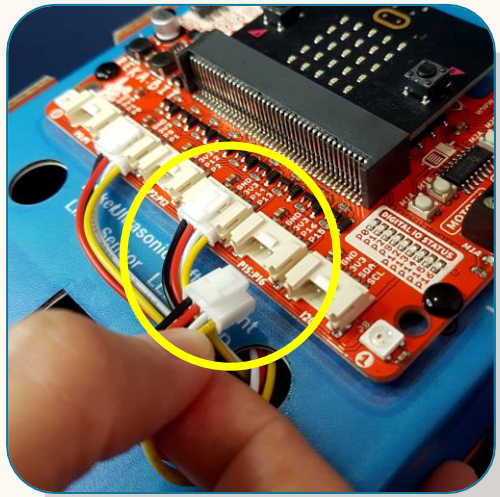
17 根据所显示的图案，使用两片双面胶把 LED 模块贴在盒子的前方。

电线连接方式：

Grove 电线	LED 模块
白	*不需要连接*
黄	IN (Input)
黑	GND (Ground)
红	VCC (Power input)







18 把左和右的 LED 模块电线穿入所标记的洞孔。

19 把**左**边的 LED 模块电线连接到 REKA:BIT 上 **P13:P14** 接口。

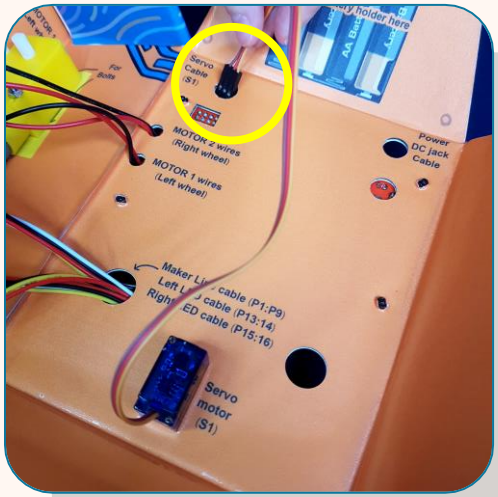
20 把**右**边的 LED 模块电线连接到 REKA:BIT 上 **P15:P16** 接口。

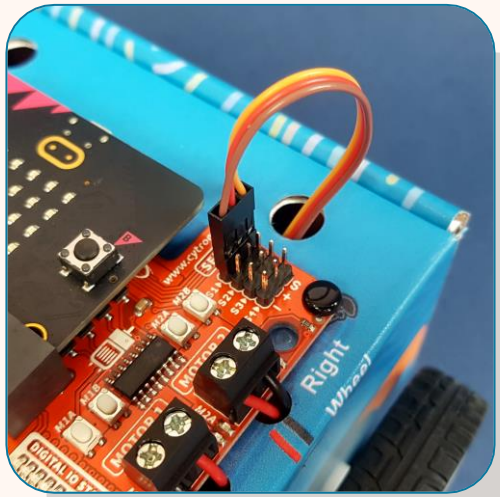
电线连接方式：

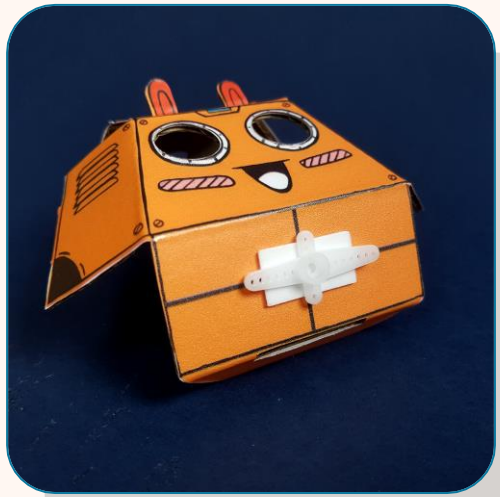
LED 模块	REKA:BIT 接口
左	P13:P14
右	P15:P16

21 使用一共四个白色铆钉，一边两个，把保险杠 (bumper) 纸板固定在盒子上。

22 把 **伺服电机** (Servo motor) 小心地放入盒子上的开口，确保它固定在如图所指定的位置。







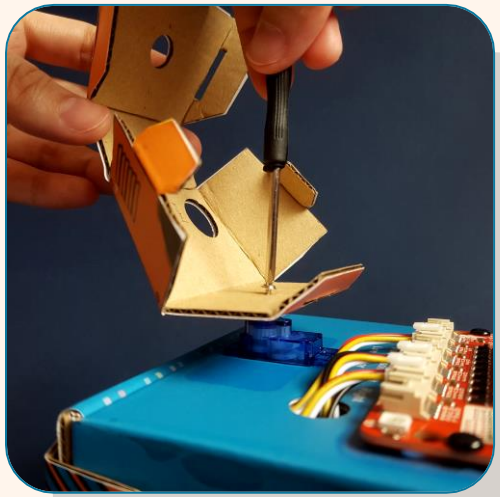
23 把伺服电机的电线穿入所标记的洞孔。

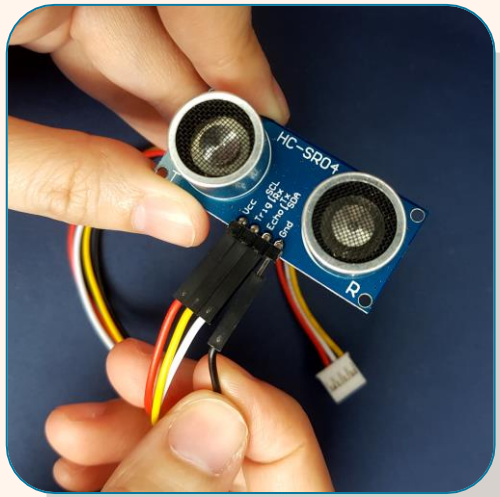
24 把伺服电机的电线连接到 REKA:BIT 上 **S1** 接口。

25 根据所显示的图案，使用一片双面胶把 **伺服电机架子** 贴在 **头部卡片** 的下方。

电线连接方式：

伺服电机 电线	Servo S1 接口
橙	S (Signal)
红	+ (Power)
褐	- (Ground)





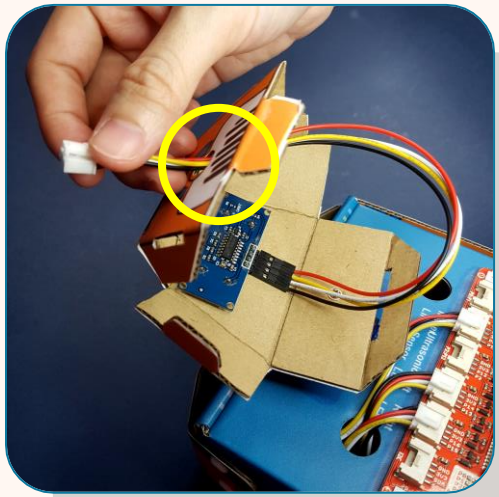
26 把**伺服电机架子**插入**伺服电机轴**上。使用所提供的螺丝刀和螺丝来固定头部的位置。

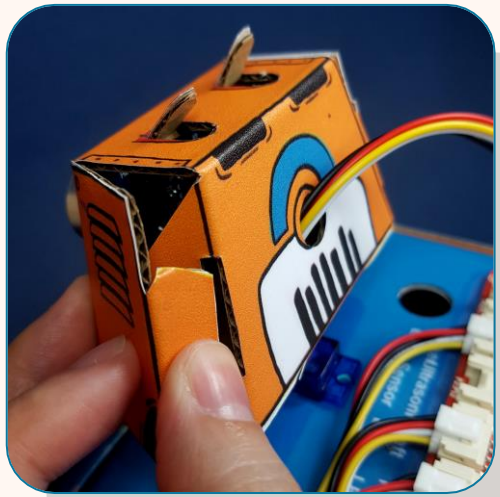
27 将一条 Grove to 4-pin female jumper 电线连接到**超声波传感器**上。

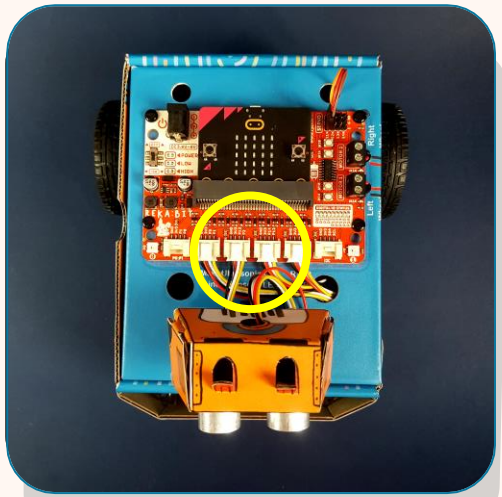
28 根据所显示的图案，把超声波传感器放入头部的卡片洞口中。

电线连接方式：

Grove 电线	超声波传感器
红	VCC (Power input)
黄	Trig (Trigger)
白	Echo (Echo)
黑	GND (Ground)



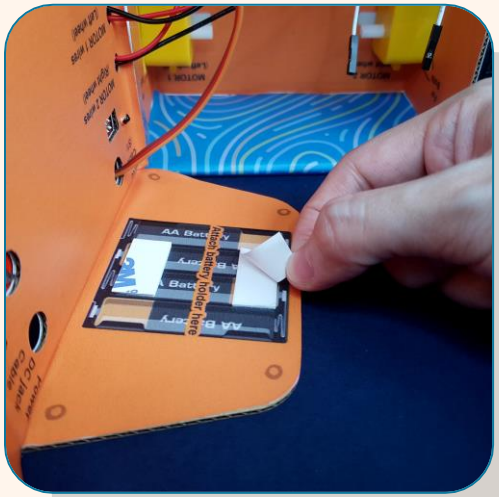


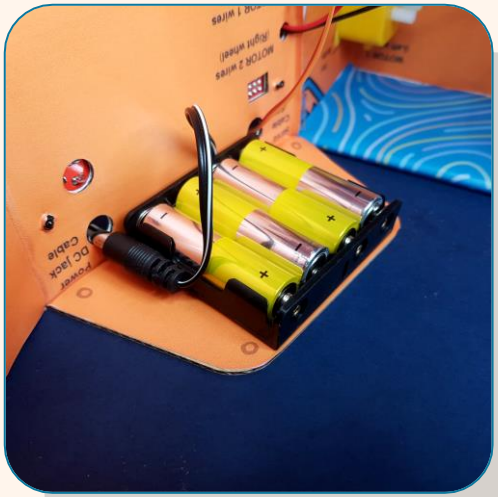


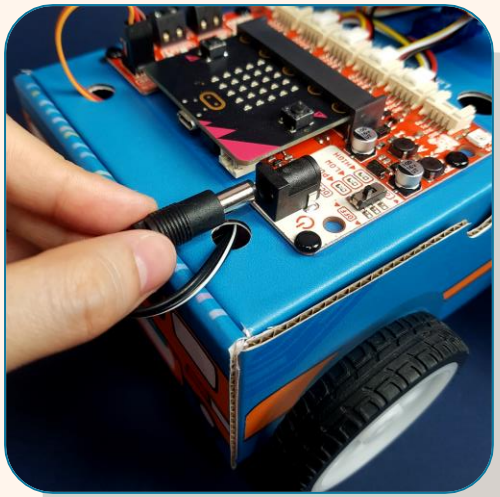
29 把超声波传感器的电线由内穿过图中所标记的洞孔。

30 跟着折线，把头部卡片折成所显示的样子，记得把襟翼 (flaps) 插入所指定的插槽。

31 超声波传感器的电线连接到 REKA:BIT 上 **P2:P12** 接口。







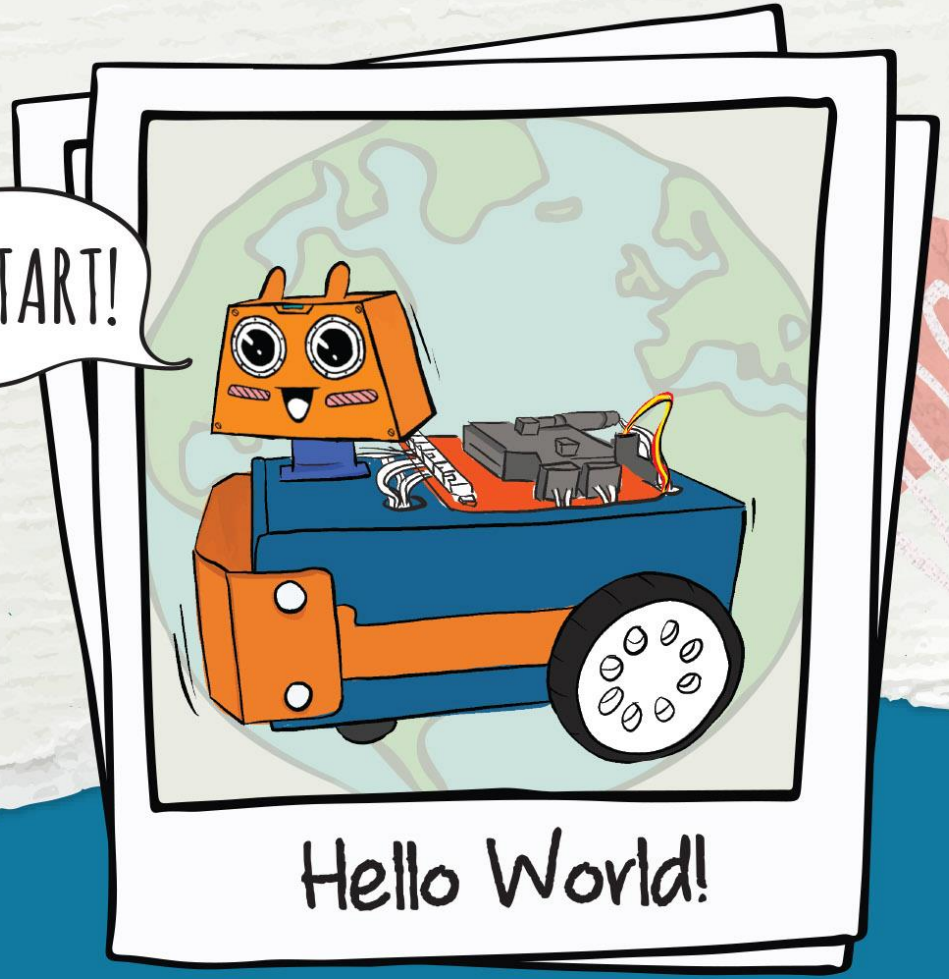
32 根据所显示的图案，使用两片双面胶把电池盒贴在 ZOOM:BIT 的盒盖侧板上。

33 把四颗 AA 电池放进电池盒。把电池盒的电线穿入所标记的洞孔。然后关上盖子。

34 把电源线连接到 REKA:BIT 的 DC 直流电源接口上。

CHAPTER 1 第一章

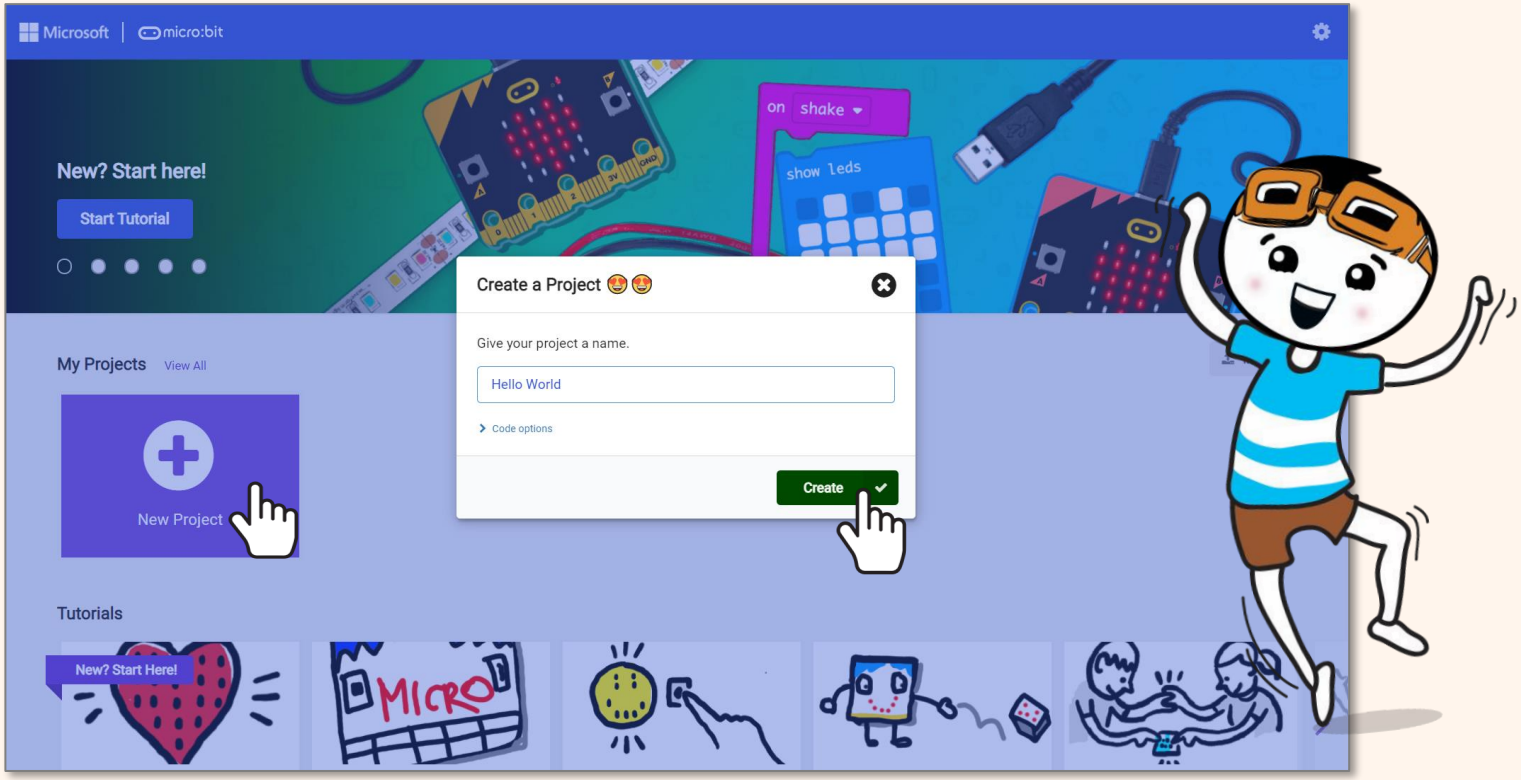
https://link.cytron.io/zoombit-chapter-1



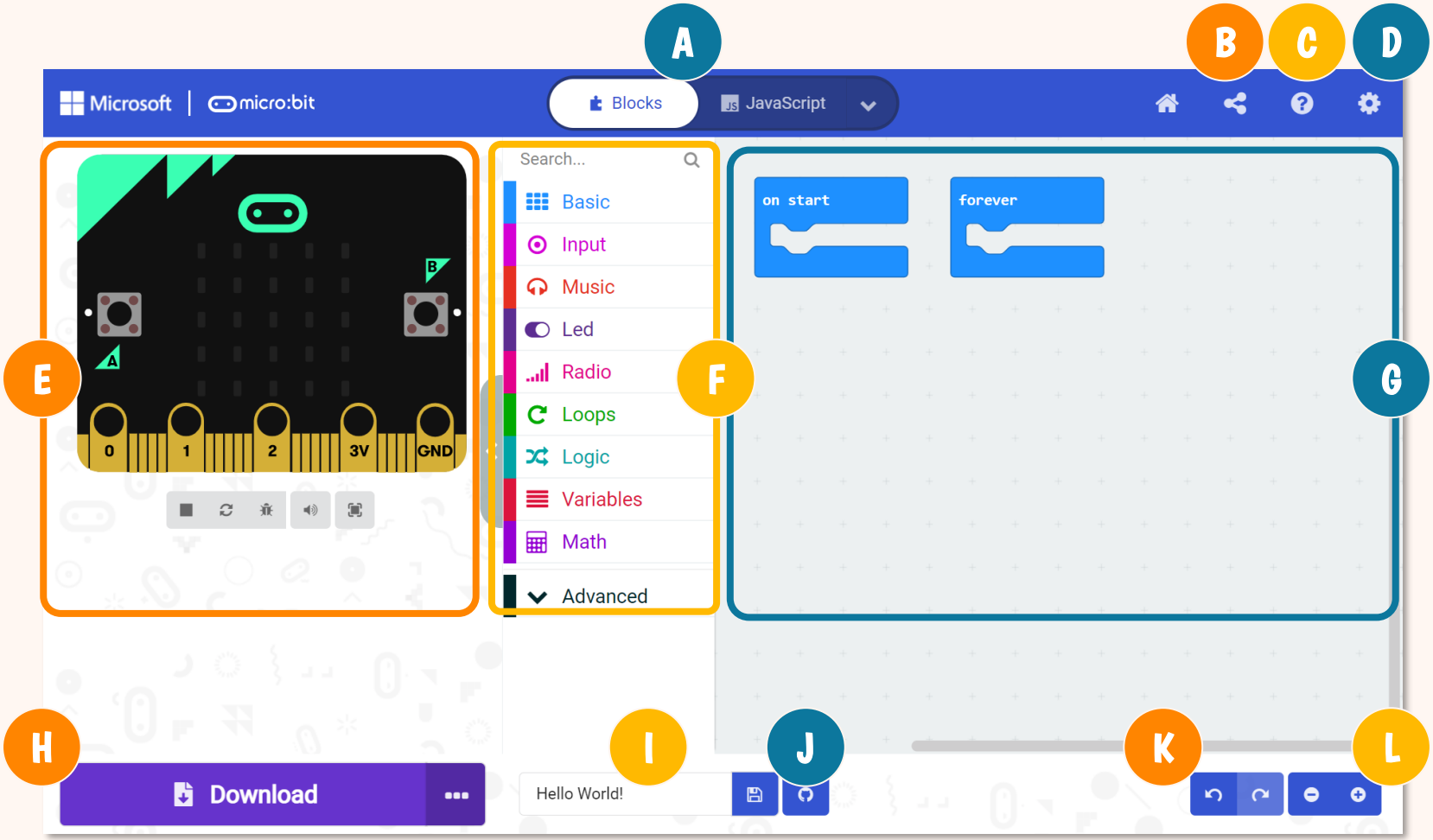
世界你好！

1 打开浏览器，输入 https://makecode.microbit.org

2 点击 [New Project]【新建项目】，输入你的项目名称 (project name)，然后点击 [Create]【创建】。



接下来，你将会看到 Microsoft MakeCode Editor 的网页，
该页面将让你使用拖放的方法，轻松续写你的程序。



A 选择你要使用 方块 Blocks，JavaScript 或 Python 来进行编程。

B 发布 publish 和 共享 share 你的项目。

C 打开帮助介面 help menu。

D 更改设置 settings，添加扩展 extensions，连接设备 connect device，等等

E **模拟器 Simulator**
– 模拟并展示出你的程序

F **工具箱 / 分类抽屉 Toolbox / Category Drawers**
– 从中获得所需的编码方块。点击以查看每一个类别里可使用的编码方块。

G **编程工作区 Programming Workspace**
– 在该区域将编码方块对齐和排列在一起来续写程序。

H 点击下载 download 你的程序至 ZOOM:BIT

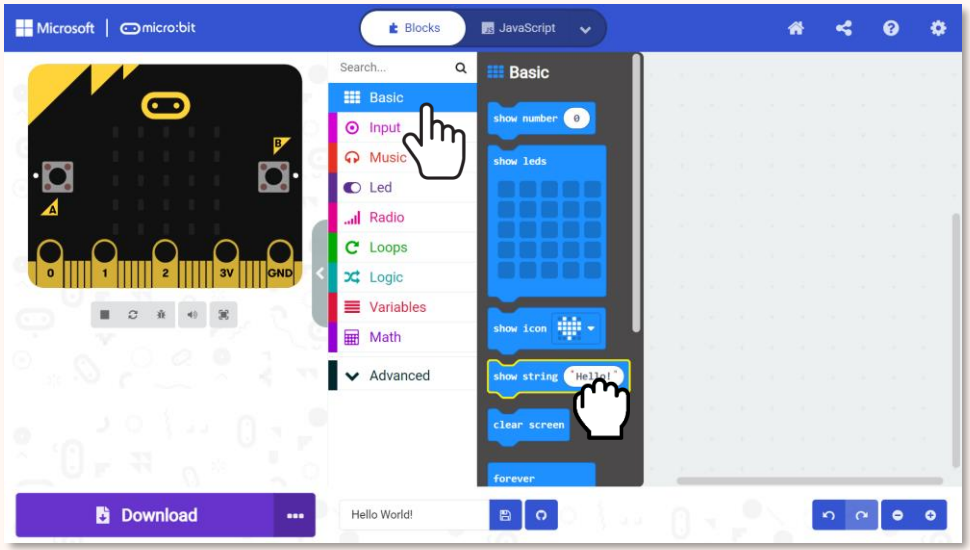
I 命名以及将你的程序保存到你的电脑。

J 创建 GitHub 资料库

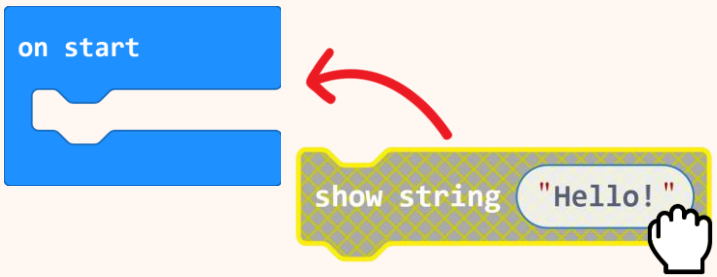
K 撤销 undo / 重做 redo

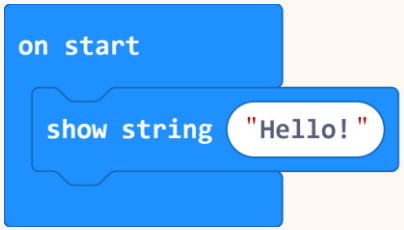
L 放大 zoom in / 缩小 zoom out

3 在工具箱点击 [Basic]【基本】，选择 [show string (“Hello!”)]【显示字符串 (“Hello!”)】的方块。

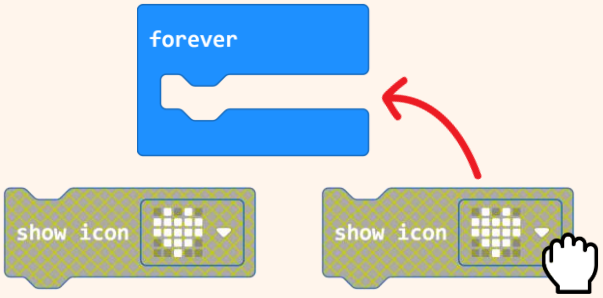


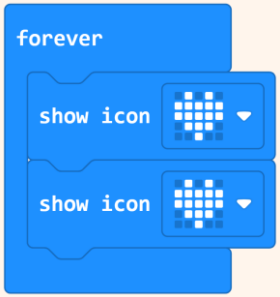
4 点击并将 [show string (“Hello!”)]【显示字符串 (“Hello!”)】的方块 排列到 [on start] 【当开机时】的插槽。



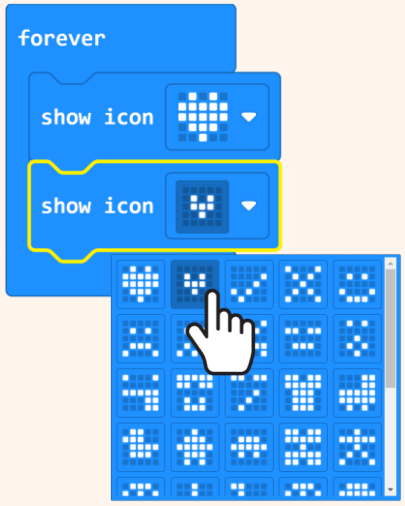


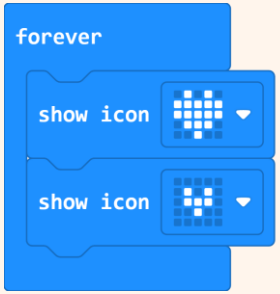
5 再点击 [Basic]【基本】，选择 [show icon]【显示图标】的方块。重复动作再创建另一个图标。选择两个 [show icon]【显示图标】的方块，把它们排列到 [forever]【无限循环】的方块里。





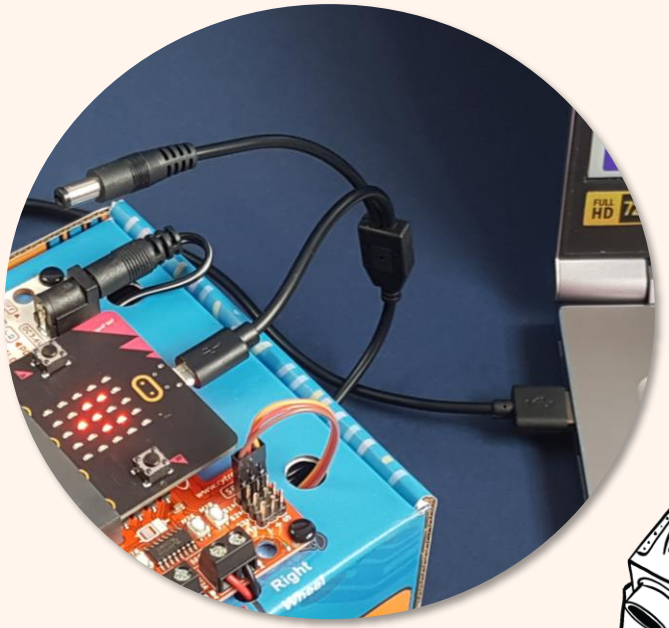
6 点击第二个 [show icon]【显示图标】方块，从弹出窗口选择 “small heart” 设计。

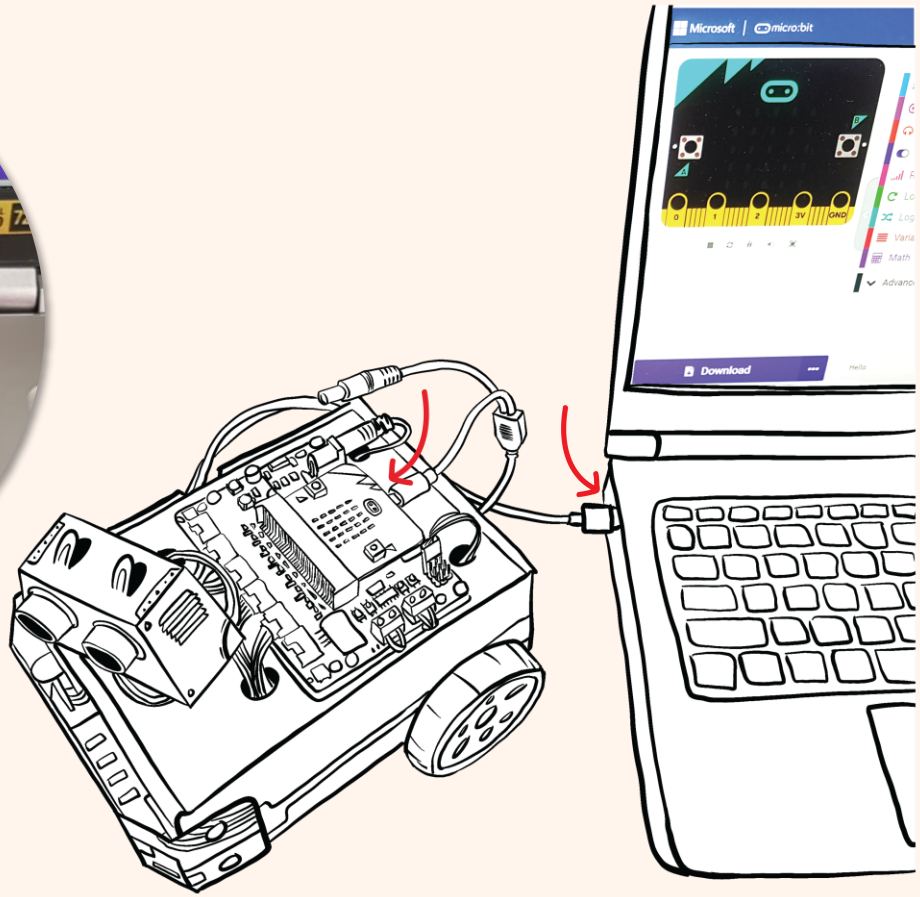




你可以在 **模拟器 Simulator** 中展示出你的程序。
在你成功下载程序后，你将会发现 “Hello!” 的字幕只出现及滚动一次，而心形图案则不断循环播放。你是否知道原因呢？

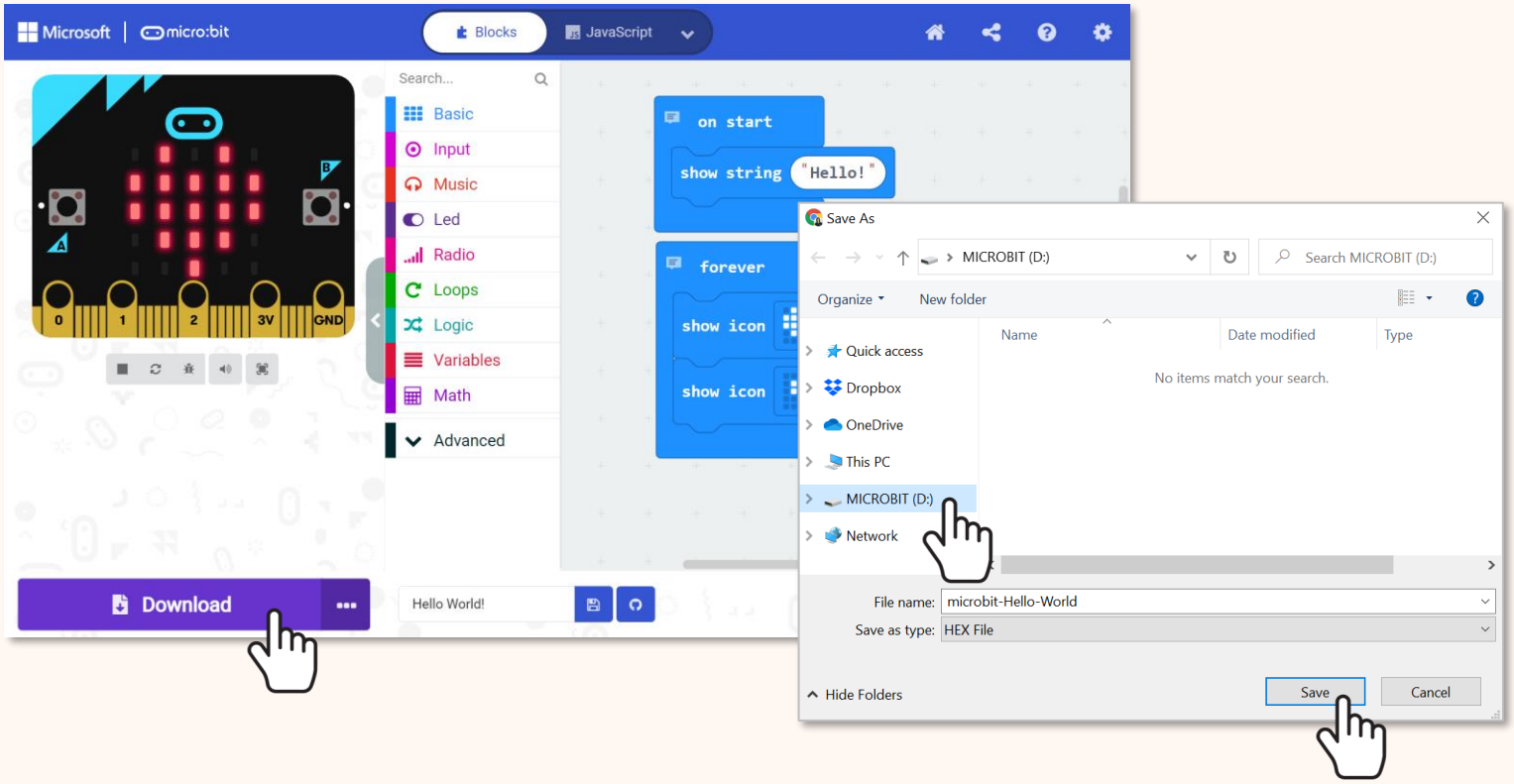
7 如图所示，将 USB **电源与数据线** 连接到你的电脑 和 micro:bit 上。



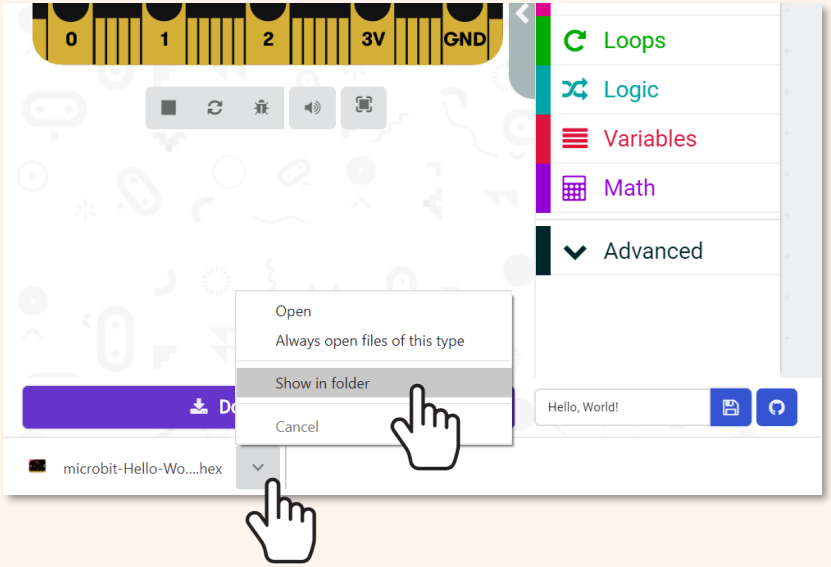


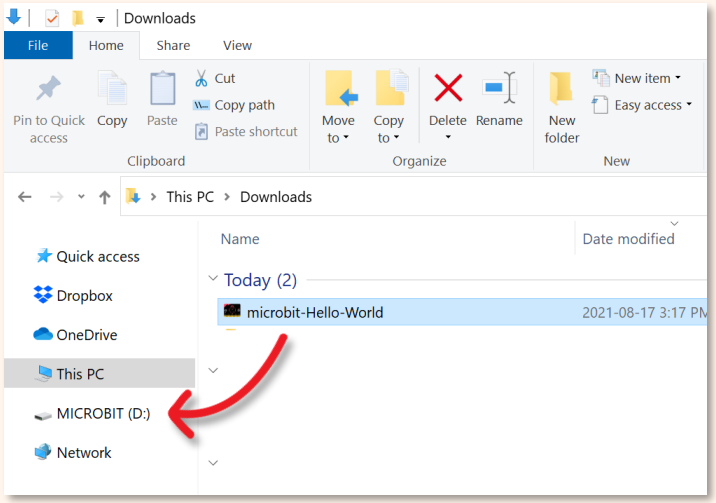
8 点击 [download]【下载】按钮。在弹出的窗口，选择将项目 (project) 下载到 MICRO:BIT 的驱动器 (drive)，然后点击 [save]【保存】。

9 当显示 “Download completed 下载完毕” 时，点击 [Done]【完毕】关闭窗口。

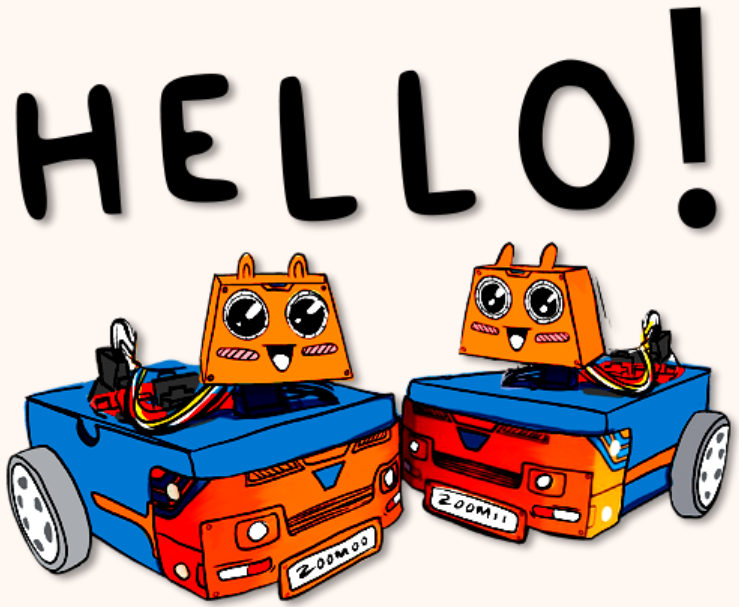


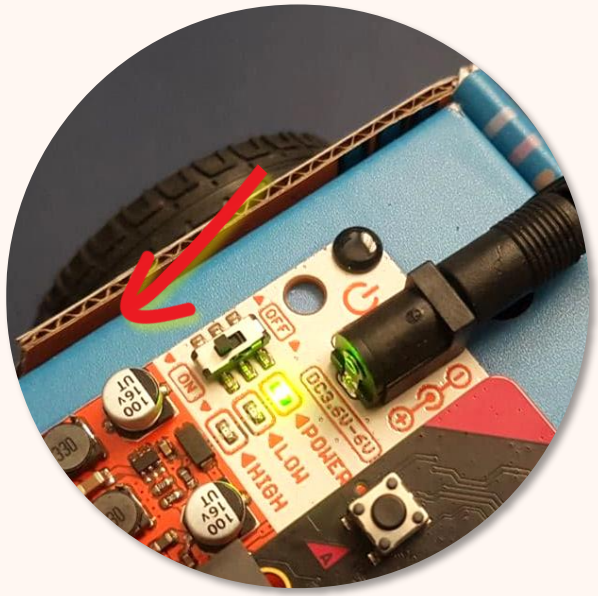
小贴士： 如果没有出现弹出窗口，则表示文件已自动下载到浏览器设置为保存下载的位置。右键单击成功下载并出现在窗口底部的 .hex 文件，然后选择 “Show in folder”。单击并拖拉已下载的 “microbit-xxxx.hex” 文件到 MICROBIT 驱动器，就像你将文件复制到闪存驱动器一样。





10 拔掉 USB 数据线的连接，滑动电源开关键到 ON 以打开 ZOOM:BIT 的电源。

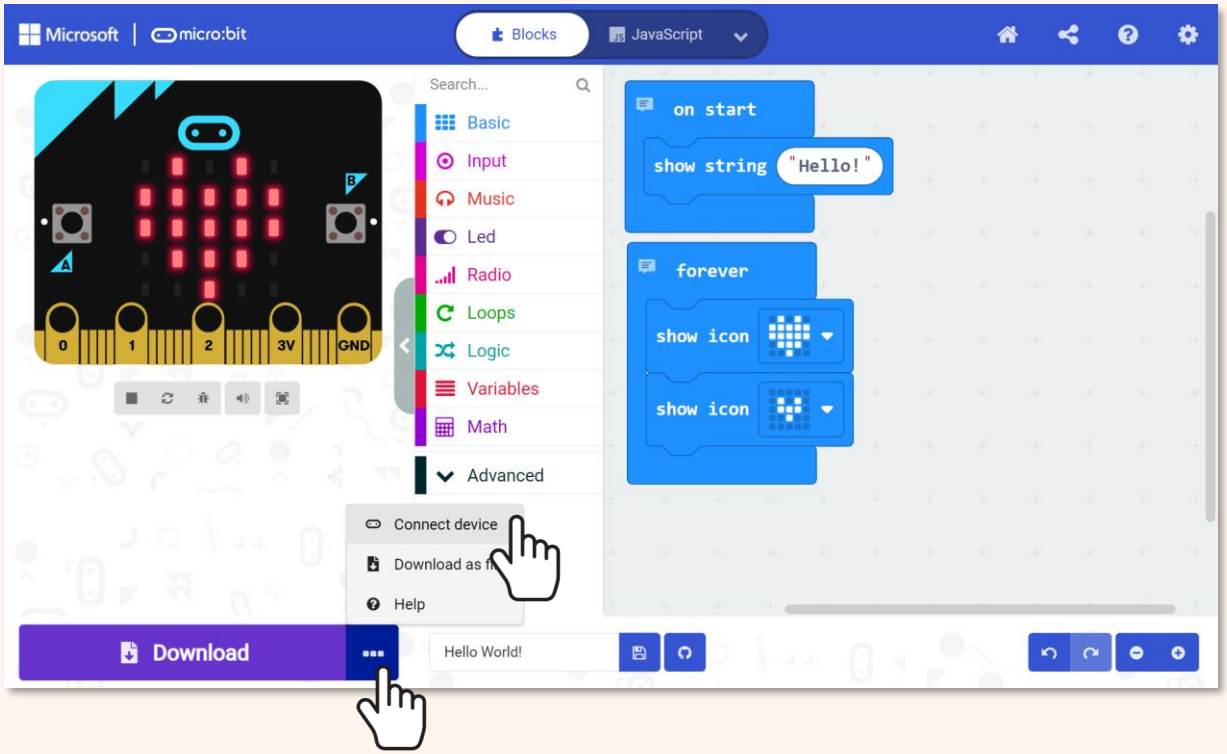




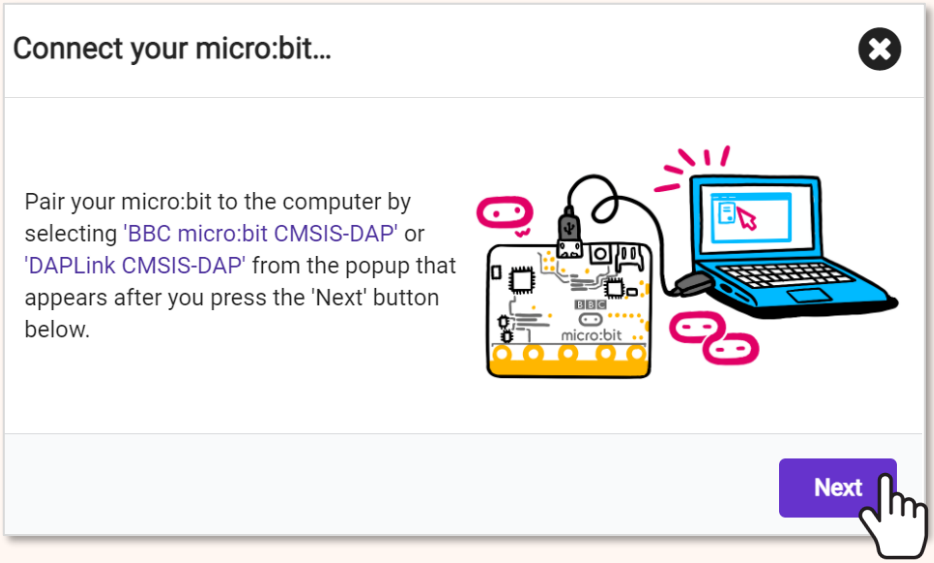
你是否有看到 “Hello!” 的字幕在 LED 矩阵 (LED Matrix) 上滚动，紧接着是心形图案的循环播放？如果你想再看多一次，你可以重启 REKA:BIT。把电源开关键滑动到 OFF，然后再重新打开电源 ON，就可以啦！

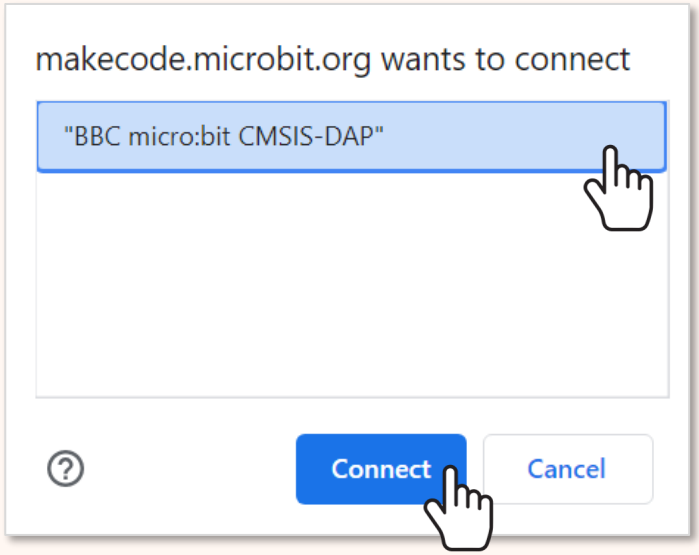
你是否知道？ 其实，你可以选择 **“connect device 设备配对”** 功能来使下载动作更简单。当你连接上你的设备后，你只需要一键点击，就能马上 **下载 (flash)** 你的程序到 ZOOM:BIT。

11 连接你的 ZOOM:BIT 到电脑。点击 **[Download]【下载】**按钮左边的**三个点按钮**，然后选择 **[Connect device]**。



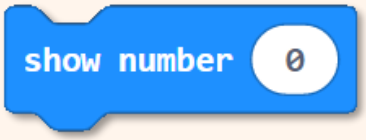
12 跟着屏幕上的指示，从列表中选择 ‘BBC micro:bit CMSIS-DAP’ 或者 ‘DAPLink CMSIS-DAP’，然后点击 [Connect]。



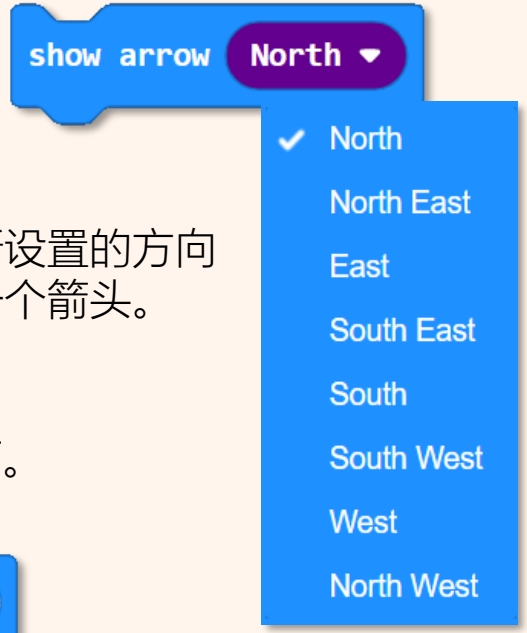


小贴士：你需要使用最新版本的 Edge 或 Chrome 浏览器，并在 micro:bit 设备上拥有最新的固件 (firmware) 。如果在连接设备时出现问题，你可以查询 https://makecode.microbit.org/device/usb/webusb/troubleshoot 以获得更多资讯。

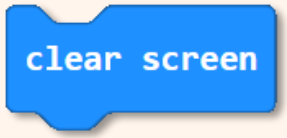
探索更多编码块



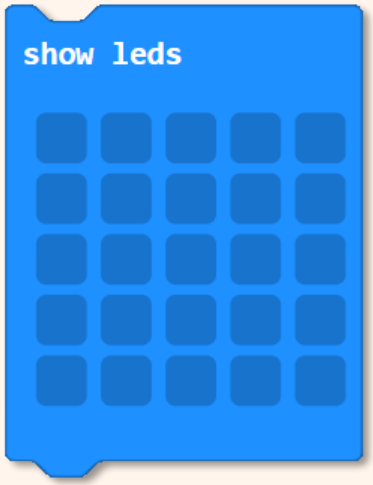
显示数字而已。



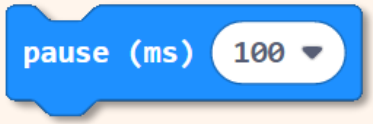
根据所设置的方向
显示一个箭头。



关闭所有 LED 灯。



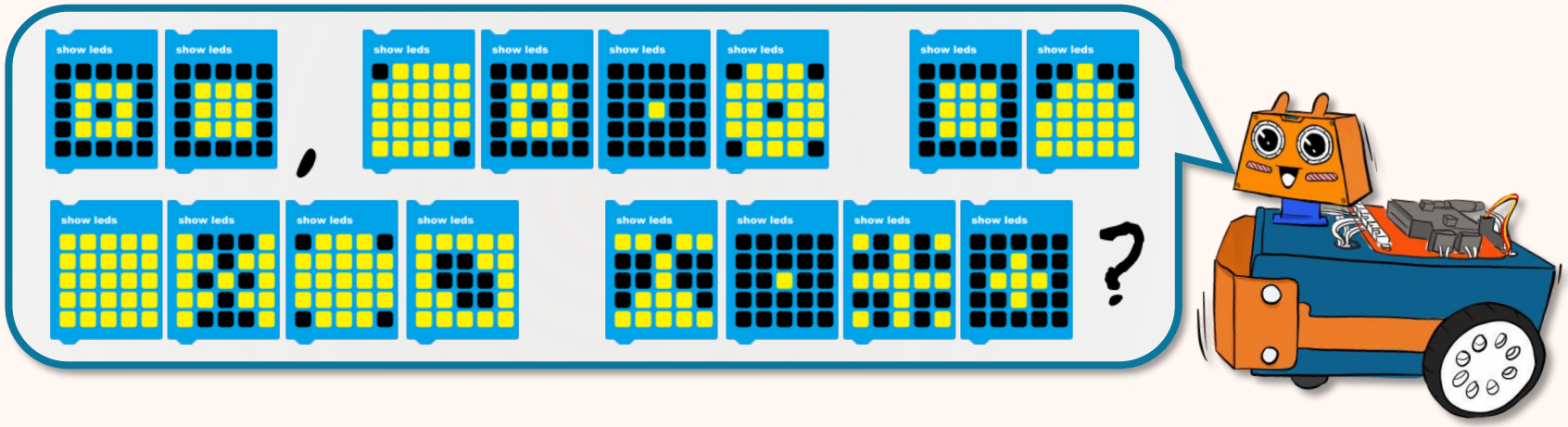
设计自己的图标来显示在 LED 矩阵
(LED Matrix) 上。
点击圆边方格来 **打开** 或 **关闭** 该 LED 灯。



添加延迟功能以减慢程序的速度。
此编码块会根据所设置的时间 (毫秒)
来暂停程序。
1000 **毫秒** milliseconds = 1 **秒** second

为你准备了一个有趣的挑战！

你能将 ZOOM:BIT 所问的问题进行解码吗？
也尝试使用相同的密码，续写一个程序让你的智能小车回复吧！



希望这些线索能帮到你~~

CHAPTER 2 第二章

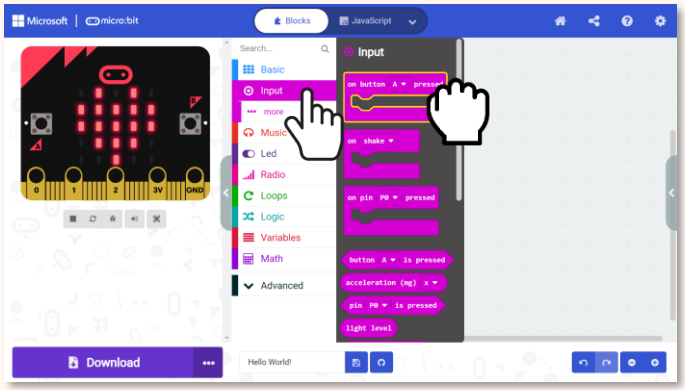
https://link.cytron.io
/zoombit-chapter-2

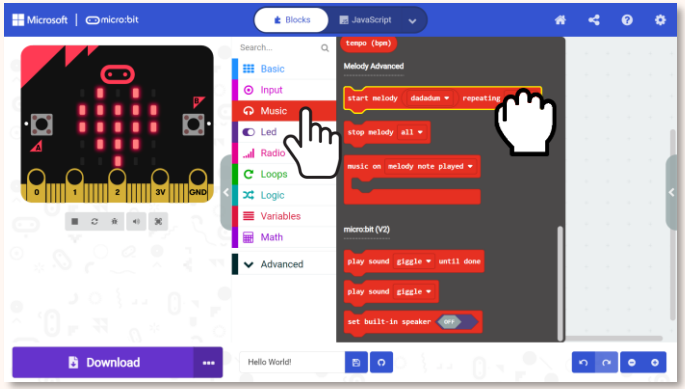
让我们听听你的歌声 ~

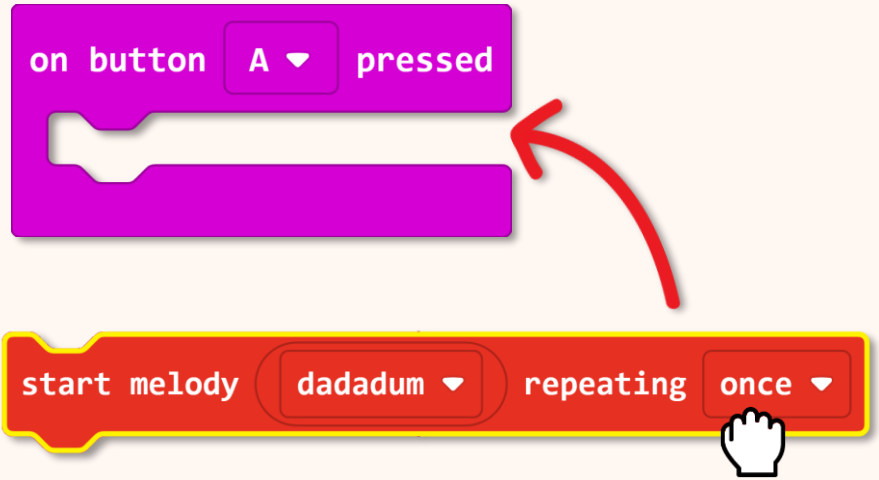
1 单击 [Input]【输入】类别，然后选择 [on button (A) pressed]【当按钮 __ 被按下时】编码块。

2 单击 [Music]【音乐】类别，然后选择 [start melody (dadadum) repeating (once)]【播放旋律 __ 重复 __ 】编码块。

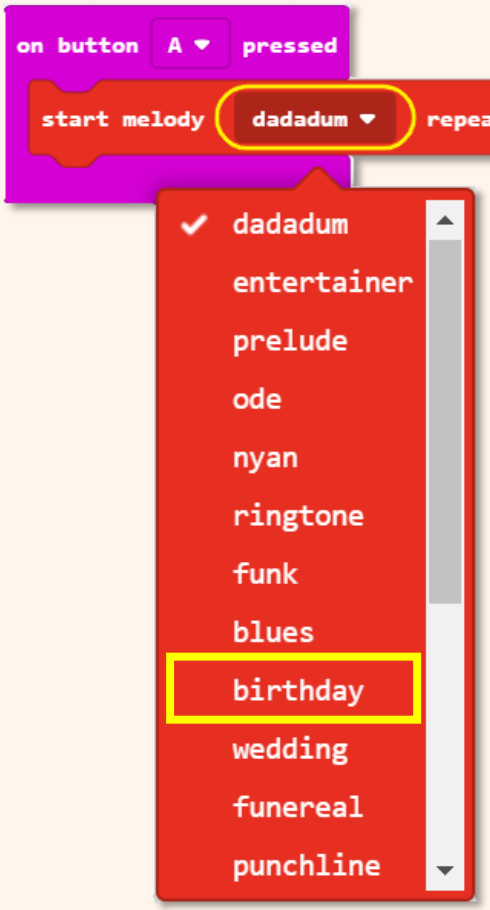
来教教 ZOOM:BIT 唱 Do Re Mi ~
你可以创建一个新项目或者继续在同样一个项目添加新的编码块。

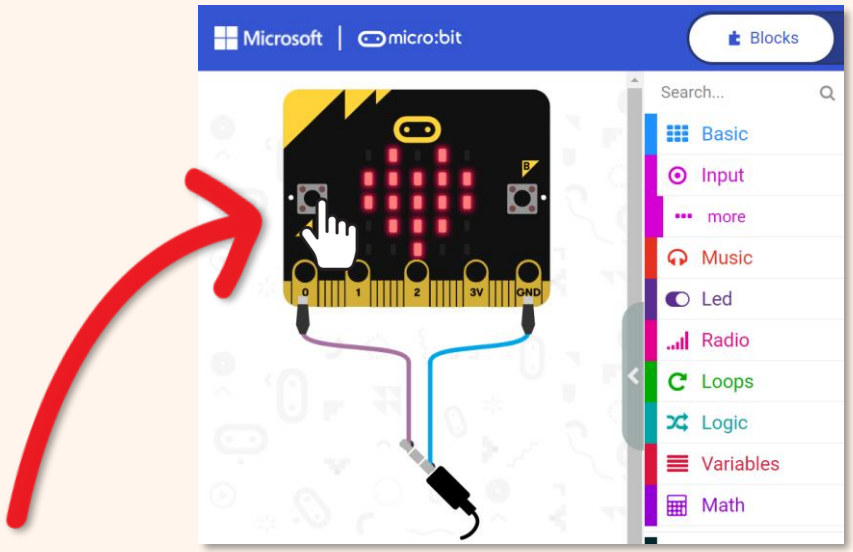






3 单击 [dadadum]，然后从下拉列表中选择 **“birthday 生日歌”** 旋律。



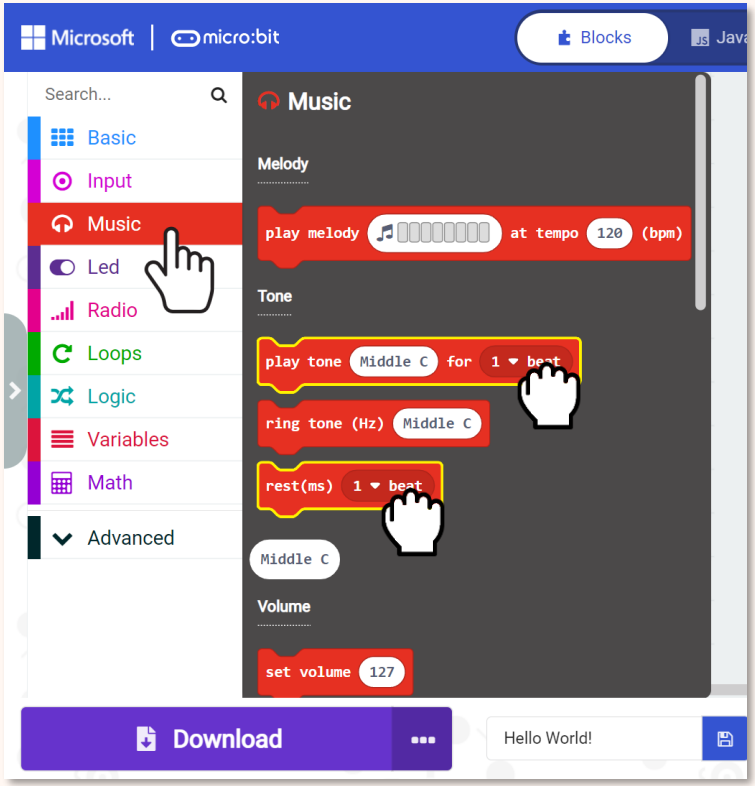


单击屏幕模拟器上的 按钮 A (Button A)。
你是否有听到一个熟悉的曲调?
也尝试其他旋律吧~

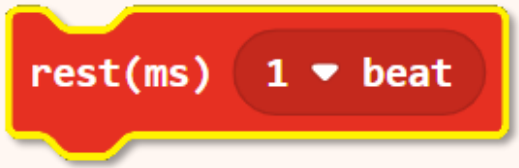
小贴士：

*请确保你的电脑扬声器已经打开。

你是否知道？ 除了预设旋律的列表外,你也可以对 ZOOM:BIT 进行编程以播放任何喜欢的歌曲。但是，你需要使用 [Music]【音乐】类别的 [play tone (middle C) for (1 beat)]【播放音调 (中C) 持续 (1节拍) 节拍】和 [rest (ms) (1 beat)]【暂停播放 (ms) (1节拍) 】来续写程序。

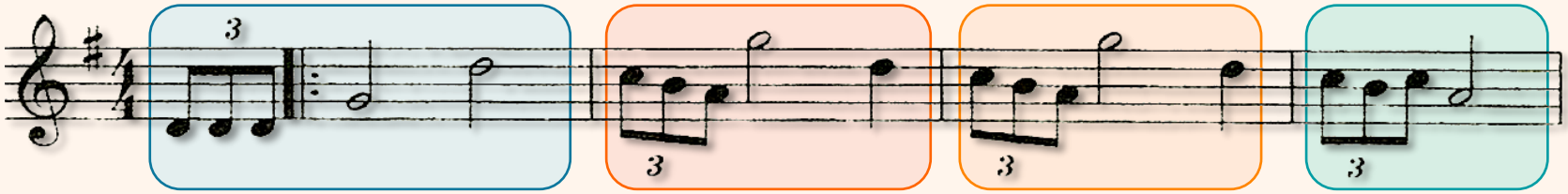


要播放的**音调 (note)**；和播放的**节拍 (时长)**



暂停，在指定的节拍内不会播放任何音调。

来一起续写 STAR WARS 主题曲的前奏的程序让 ZOOM:BIT 播放吧~~



音调	Middle D	Middle D	Middle D	Middle G	High D
节拍	1/3	1/3	1/3	2	2

音调	High C	Middle B	Middle A	High G	High D
节拍	1/3	1/3	1/3	2	1

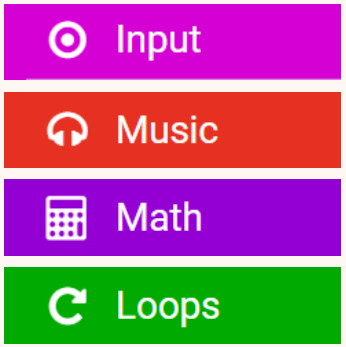
音调	High C	Middle B	Middle A	High G	High D
节拍	1/3	1/3	1/3	2	1

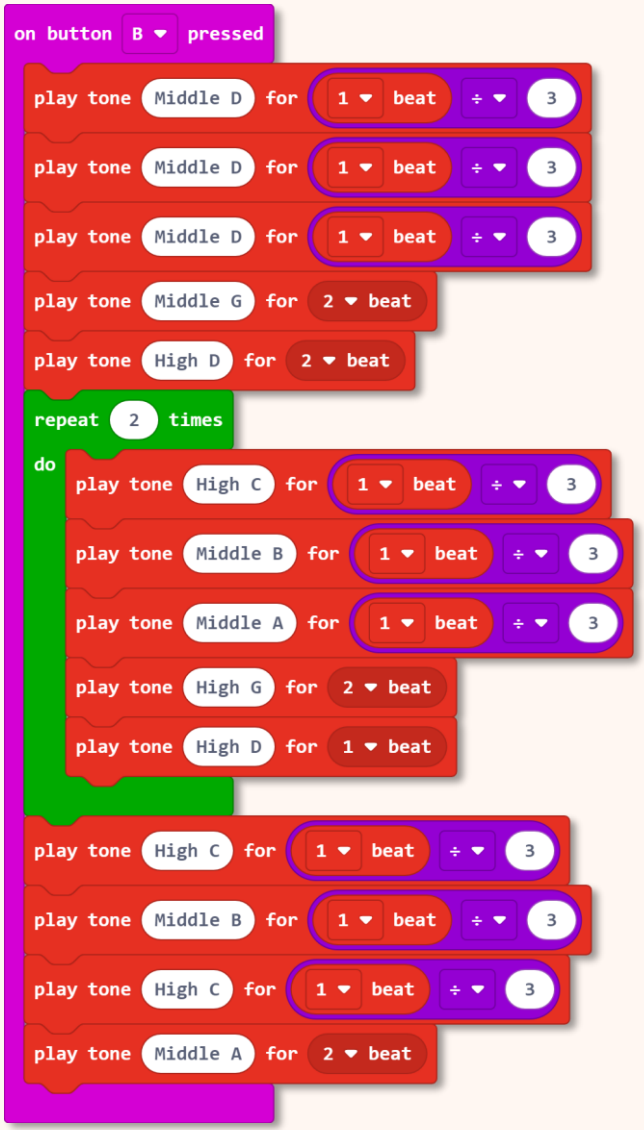
这两行的音符是一样的。你可以使用 **重复** Loop 的方块来使你的程序更简单易读。

音调	High C	Middle B	High C	Middle A
节拍	1/3	1/3	1/3	2

4

把右边的全部编码块续写到你的程序里。你将发现那些编码块可以从一样颜色的**分类抽屉** category drawers 获得。



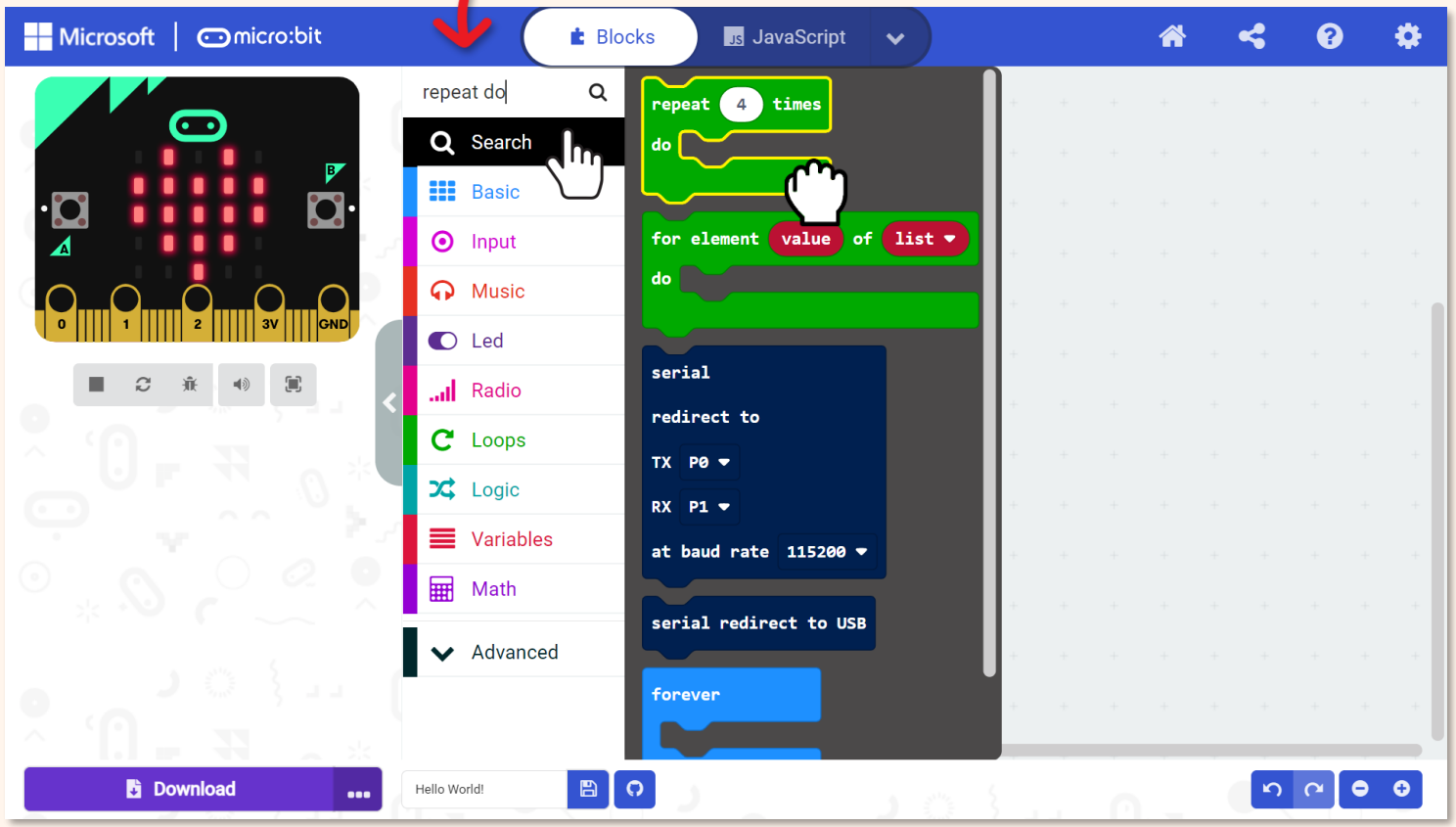


你是否知道？

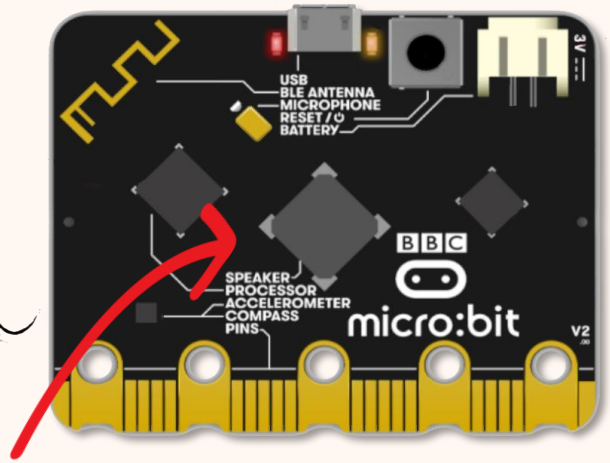
所有的编码块都是有颜色标记的。你可以从一样颜色的分类抽屉获得你需要用到的编码块。

如果你需要更多指导，你可以浏览 https://link.cytron.io/zoombit-tutorial-2 获取更多分步指南。

你是否知道？ 你也可以在搜索栏输入关键字，以获得你所需要的编码块。



5 下载 (flash) 你所完成的程序到 ZOOM:BIT。打开电源开关，然后按下 按钮 B 。



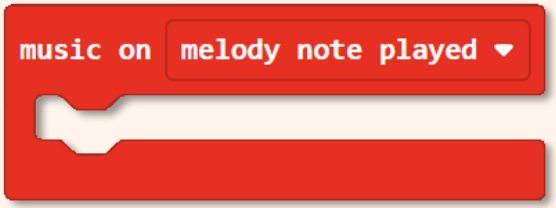
你的 ZOOM:BIT（如果是 micro:bit 第二代）将可以“唱歌”和播放音乐，因为它拥有内置扬声器 built-in speaker，使它能够产生声音。

小贴士：

如果你使用着 micro:bit 第一代，它是没有内置扬声器的，你将需要连接一个 Grove buzzer 到 P0:P1 接口，才能够播放音乐和产生声音。你可以浏览 https://link.cytron.io/zoombit-grove-buzzer 或扫描右边的二维码来了解更多。

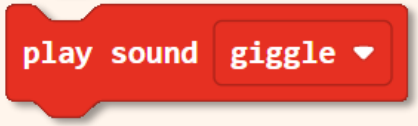
探索更多编码块

使用旋律编辑器，
根据所设置的拍子
撰写和播放短旋律

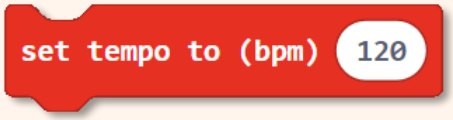


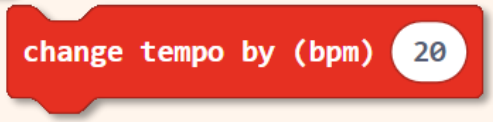
使用条件程序，例如旋律开始和旋律结束时，
作为代码中的触发事件。

停止当前正在播放和
待播放的所有旋律

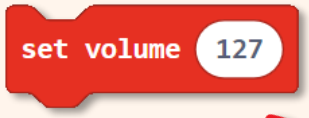


表达出各种声音
(只限于 micro:bit V2)。

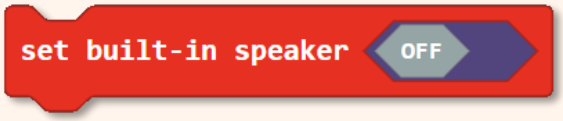




设置 set 和 更改 change “速度”
(即歌曲的步调)。每分钟的节拍
(bpm) 越高，音乐会越快或更活泼。

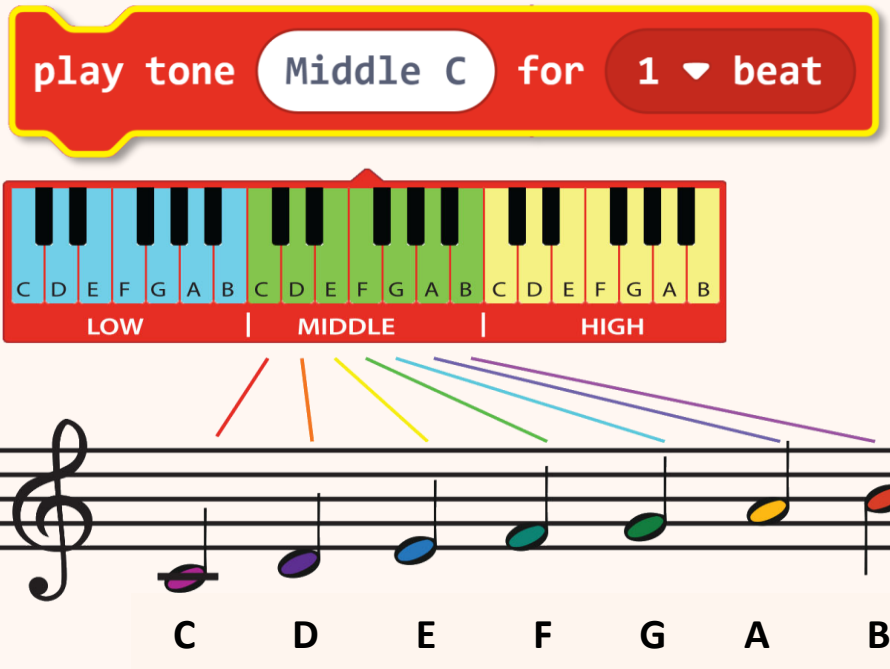


调节音乐的响度，
范围从 0 至 255
(255是最大声)。



启用或禁用内置扬声器来播放
声音 (只限于 micro:bit V2)。

如果你能够理解乐谱，你可以自行续写一首曲子的程序让 ZOOM:BIT 播放。
以下是一个简单的指南来帮助你剖析乐谱。



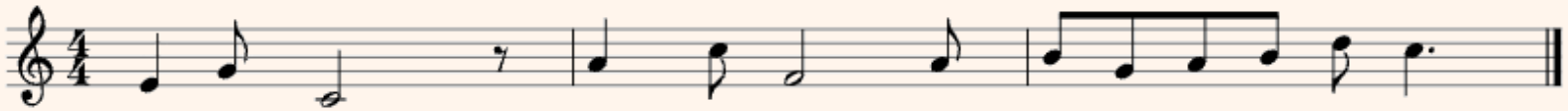
音符在谱表 (例：五线谱) 上的位置，
告诉我们需要播放的音调。
音符在五线谱上的位置越高，
音调和频率就越高，反之亦然。

音符	休息	时长
𝅝	𝄻	4 节拍
𝅗𝅥	𝄼	2 节拍
♩	𝄽 or 𝄾	1 节拍
♪	𝄿	1/2 节拍
𝅘𝅥𝅯	𝅀	1/4 节拍

不一样的符号告诉
我们每一个音符的
播放时长。

为你准备了一个有趣的挑战！

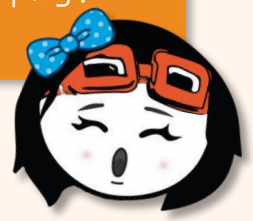
来教导 ZOOM:BIT “唱” 你最喜爱的歌吧。 你将需要续写每一个音符和节拍的程序。
如果你暂时没有想到任何歌曲，你可以尝试以下曲子 :-



音调	Middle E	Middle G	Middle C	Rest	Middle A	High C	Middle F	Middle A
节拍	1	1/2	2	1/2	1	1/2	2	1/2

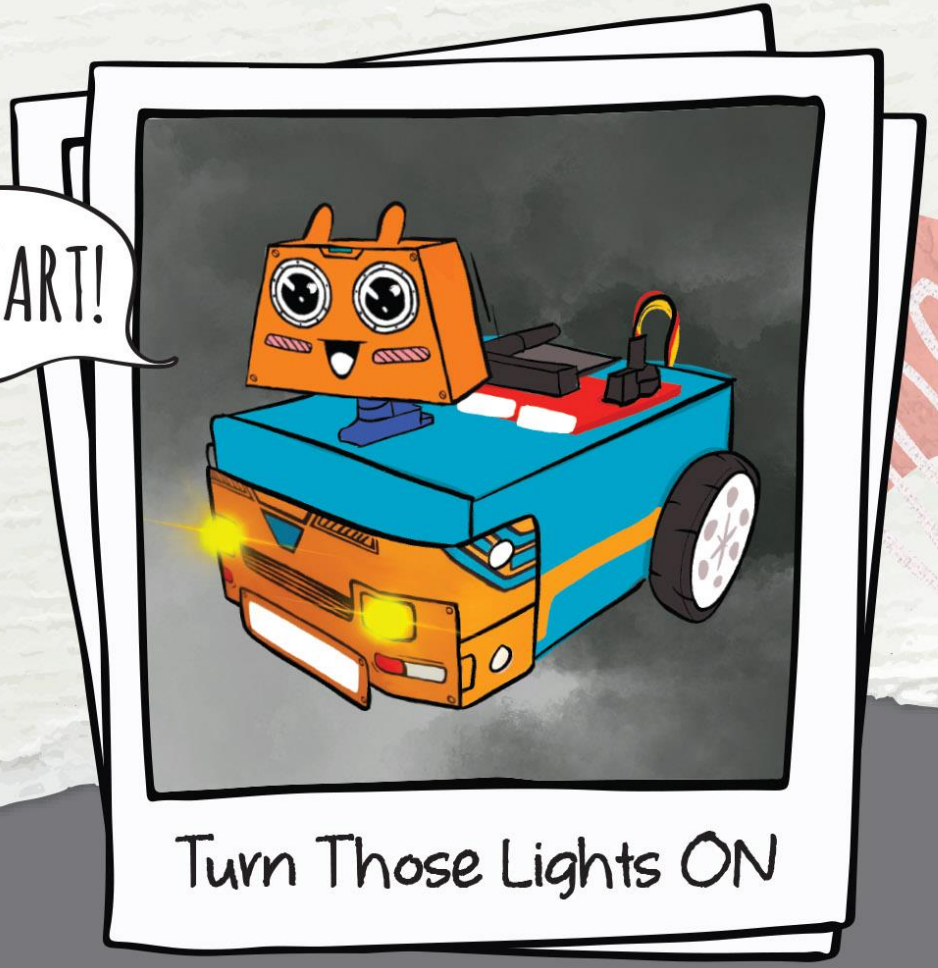
音调	Middle B	Middle G	Middle A	Middle B	High D	High C
节拍	1/2	1/2	1/2	1/2	1/2	1 1/2

这是一段耳熟能详的曲调。你能猜到是什么旋律吗?



CHAPTER 3 第三章

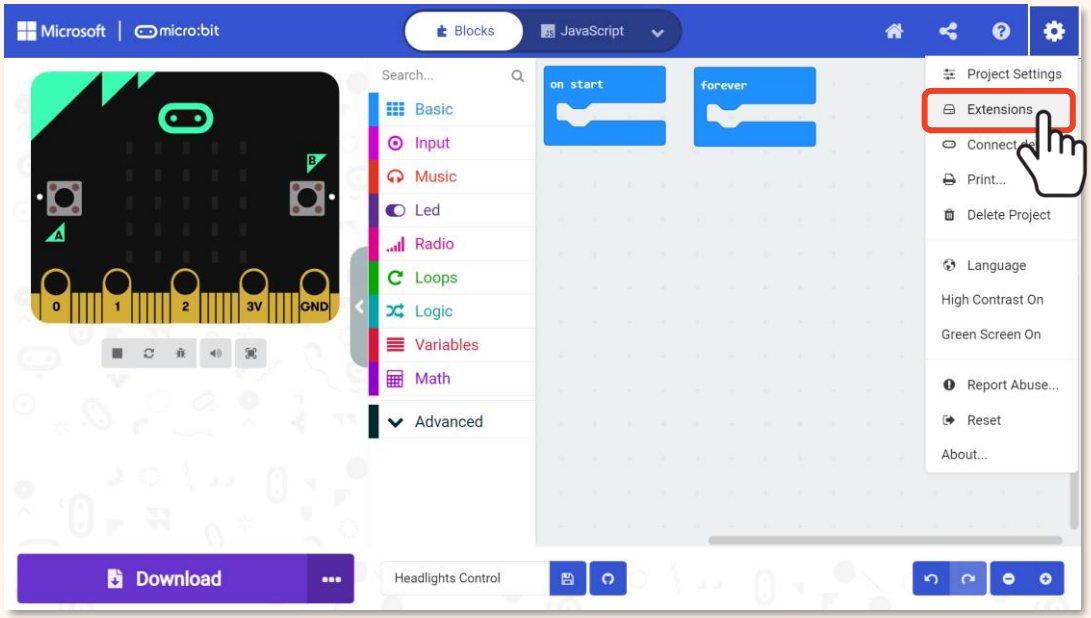
https://link.cytron.io/zoombit-chapter-3



打开车前灯

你是否知道？ micro:bit 上的 LED 矩阵 (LED Matrix) 也有灯光亮度感应的功能。让我们一起续写一个程序让 ZOOM:BIT 在周围变暗时自动打开车前灯；在周围变亮时则自动关掉车前灯。

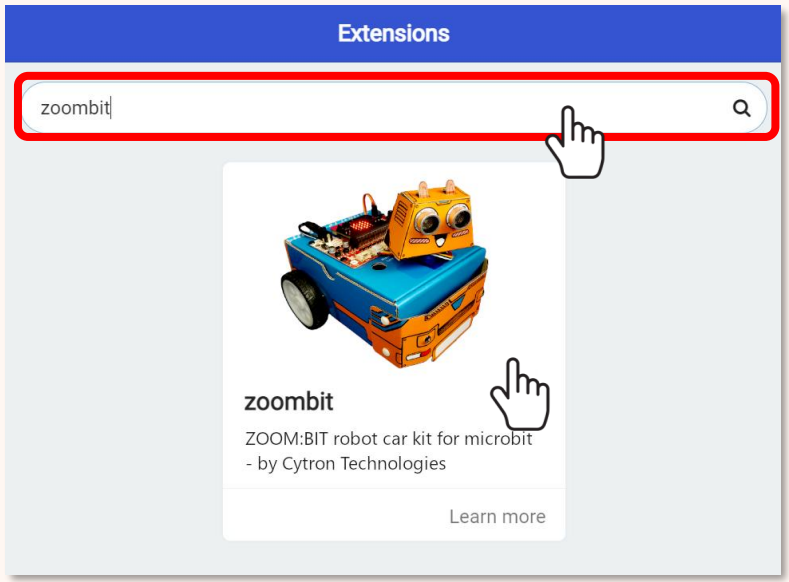
1 在你的 MakeCode Editor 中创建一个新项目。 单击齿轮图标  ，然后选择 **"Extensions 扩展"**。 **你需要网络连接才能可以添加扩展。**

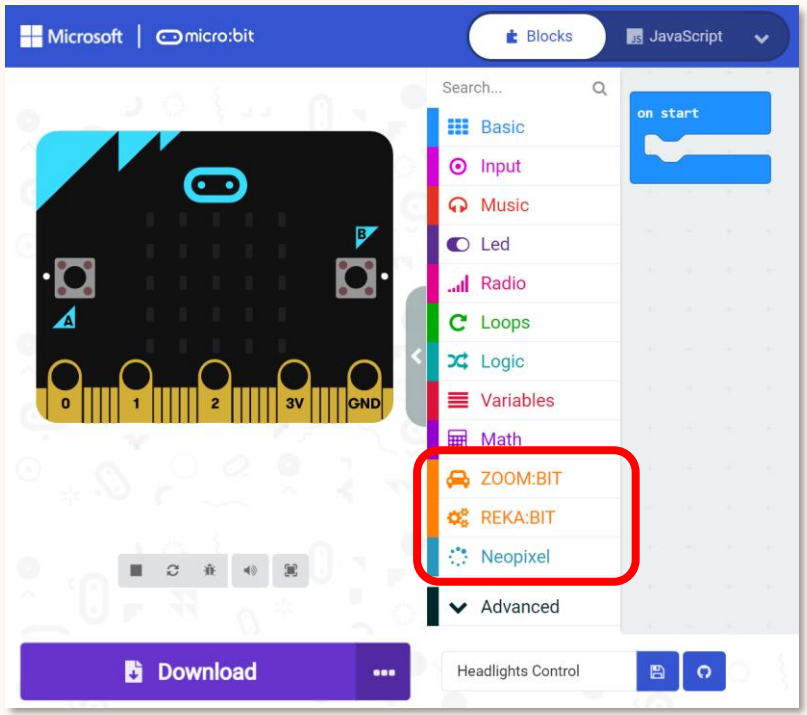


扩展 Extensions 是我们添加到 MakeCode Editor 中的一组自定义模块，使我们能轻松地对 micro:bit 的零件进行编程，例如我们的 ZOOM:BIT 智能小车。

2 在搜索框中输入 **'zoombit'**（或者 https://github.com/CytronTechnologies/pxt-zoombit），然后单击 Enter。

3 点击 **"zoombit"** 扩展的选项。网页加载完毕后，你将在 MakeCode Editor 网页注意到如图的新类别抽屉，已被添加至 MakeCode Editor 中。





4 续写以下程序。

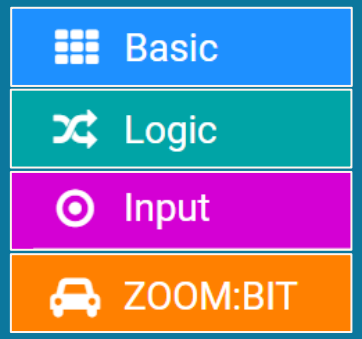
在开始时，显示心形图案

时时刻刻检查周围环境的灯光亮度

如果灯光亮度小于 50 (例如周围很暗)，就**打开**车前灯。

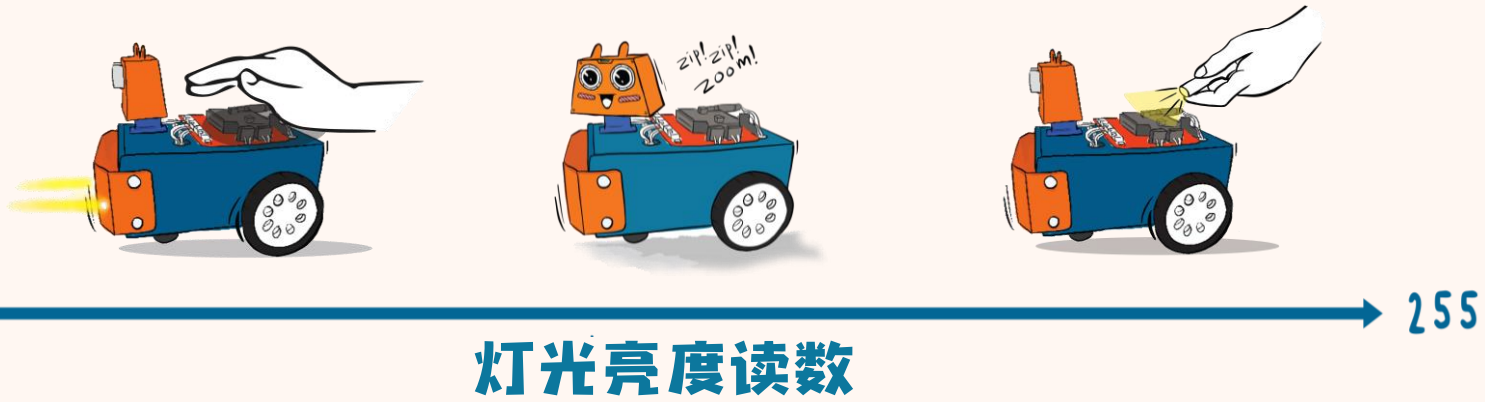
否则 (周围很亮)，**关闭**车前灯

你可以到以下**分类抽屉**寻找你所需要的编码块。



如果你需要更多指导，你可以浏览 https://link.cytron.io/zoombit-tutorial-3 获取更多分步指南。

5 把程序下载到 ZOOM:BIT，然后打开电源键。观察车前灯的情况。

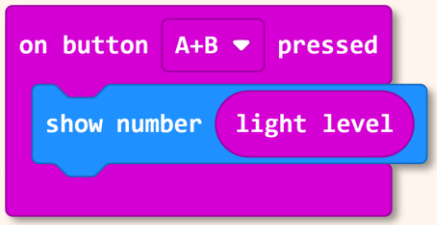


ZOOM:BIT 的车前灯是否有打开？如果没有，尝试在 LED 矩阵上制造影子。接着，使用灯光照射在 LED 矩阵上；你是否有发现任何变化？

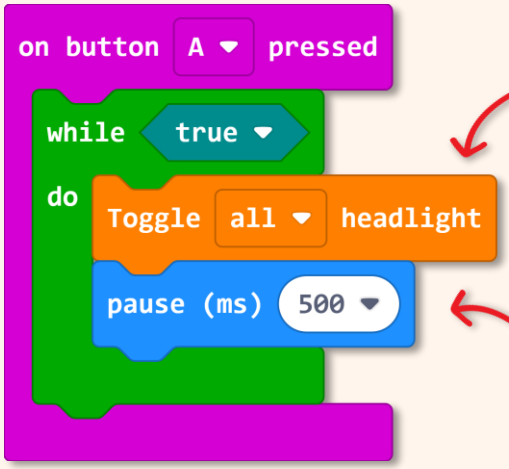
小贴士： 灯光亮度的读数是从 0 (没有灯光) 到 255 (最高亮度)。

探索更多编码块

在你的 MakeCode Editor，创建一个新的项目，续写以下的程序，然后下载到你的 ZOOM:BIT。当你同时按下按钮 A 和 B 时，是否有发现任何反应？又或是按下 按钮 A 而已呢?



在按钮 A+B 同时被按下时，
读取灯光亮度指数以及
显示读数在 LED 矩阵上。



"切换 Toggle" 是指从一种状态切换到另一种状态。
如果当前状态为 ON 打开，
则它将切换为 OFF 关闭；
反之亦然。

这个 pause 暂停方块将放慢程序的速度，才得以让你看见车前灯的 **开** ON 与 **关** OFF 的状态。

(i) 请问你当前的房间灯光亮度读数是多少？当你使用灯光照射在 LED 矩阵上时，灯光亮度读数又是多少呢?

*为了获取更准确的读数，你可以读取 3 到 4 次的指数，然后获取平均值。

(ii) 当 按钮 A 被按下后，你是否发现车前灯不断地闪烁？把电源键关闭，就能停止此程序了。

为你准备了一个有趣的挑战！

来教导 ZOOM:BIT 使用摩尔斯电码 Morse Code 交流。
续写一个程序以让 ZOOM:BIT 依据 按钮 A 或 B 被按下的情况，闪烁它的车前灯。

当 按钮 A 被按下时	**打开** 两个车前灯，维持 **500ms**，然后 **关闭**	● **点**
当 按钮 B 被按下时	**打开** 两个车前灯，维持 **1500ms**，然后 **关闭**	▬ **划**

根据所提供的国际摩尔斯电码表，你是否能够根据正确的顺序，同时按下按钮 A 和 B，发出一个 S.O.S. 的信息?

你可以点击链接观看演示视频:
https://link.cytron.io/zoombit-morse-code

国际摩尔斯电码

1. 一**点**的长度是一个单位。
2. 一**划**的长度是三个单位。
3. 在一个字母中点划之间的间隔是一点。
4. 两个字母之间的间隔是三点 (一划)。
5. 两个单词之间的间隔是七点。

CHAPTER 4 第四章

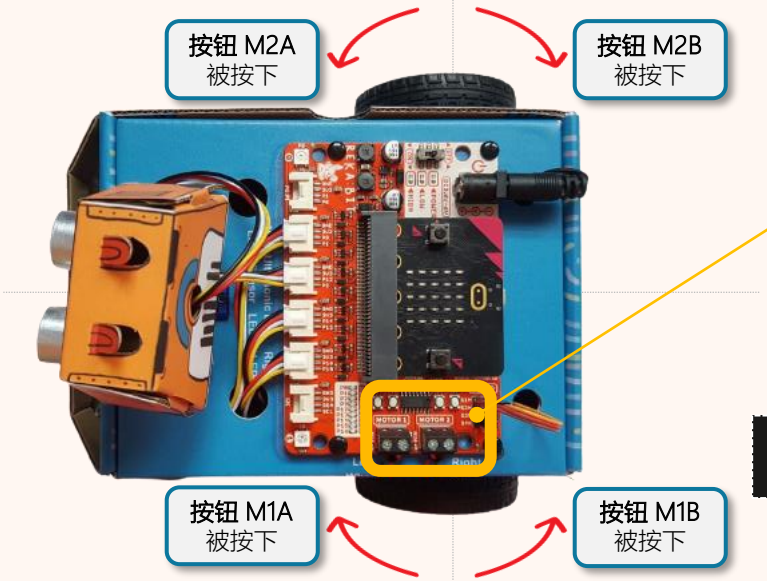
https://link.cytron.io/zoombit-chapter-4

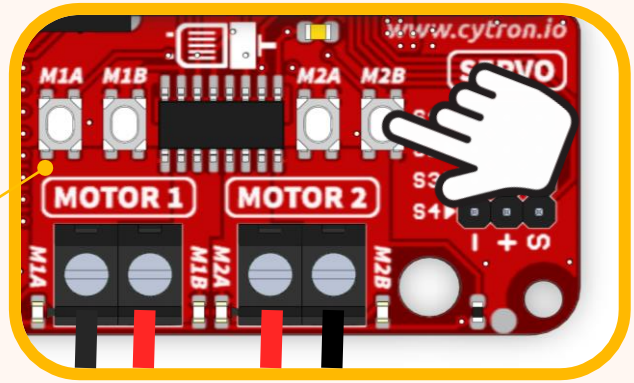
一起动起来！

在开始编程让 ZOOM:BIT 行驶前，让我们来检查所有的电线都已经正确地连接。

1. 打开电源键 ON。

2. 在 REKA:BIT 上一个接一个按下 – 按钮 M1A, M1B, M2A 以及 M2B ，然后观察轮子转动的方向。





小贴士：

如果轮子转动的方向与**红色箭头**所指的方向不一致，请检查直流马达电线的连接，并做出更改。

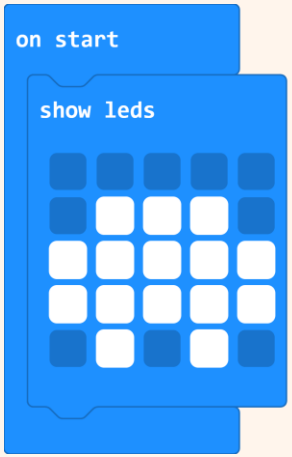
*(你可以参考第 5 和 6 页的 **电线连接方式**)*

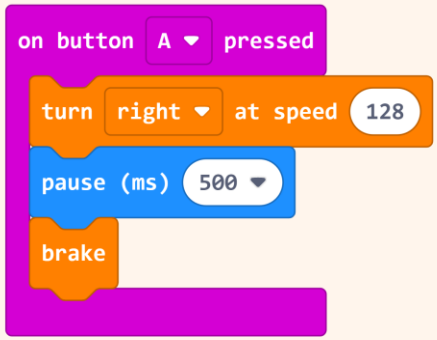
现在，我们已经准备好为 ZOOM:BIT 续写行驶的程序了。。。
让我们开始吧！ Zip zip Zoom ~

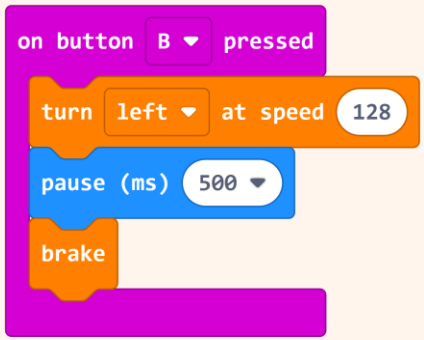
1 在你的 MakeCode Editor 中创建一个新项目。 然后添加 ZOOM:BIT 的扩展 。
(你可以参考第 44-45 页)

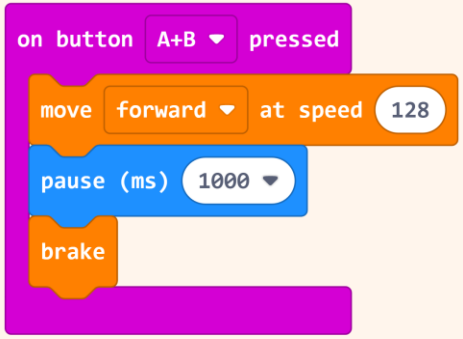
2 续写以下程序。你可以到以下**分类抽屉**寻找你所需要的编码块。

Basic | Input | ZOOM:BIT







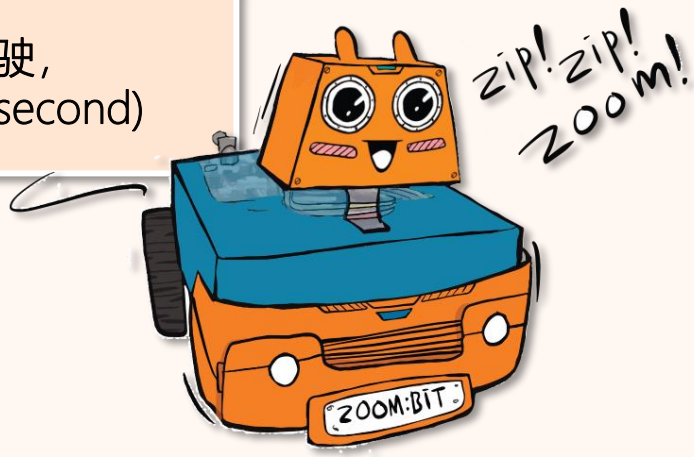


如果你需要更多指导，你可以浏览
https://link.cytron.io/zoombit-tutorial-4 获取更多分步指南。

3 把程序下载到 ZOOM:BIT，然后打开电源键。

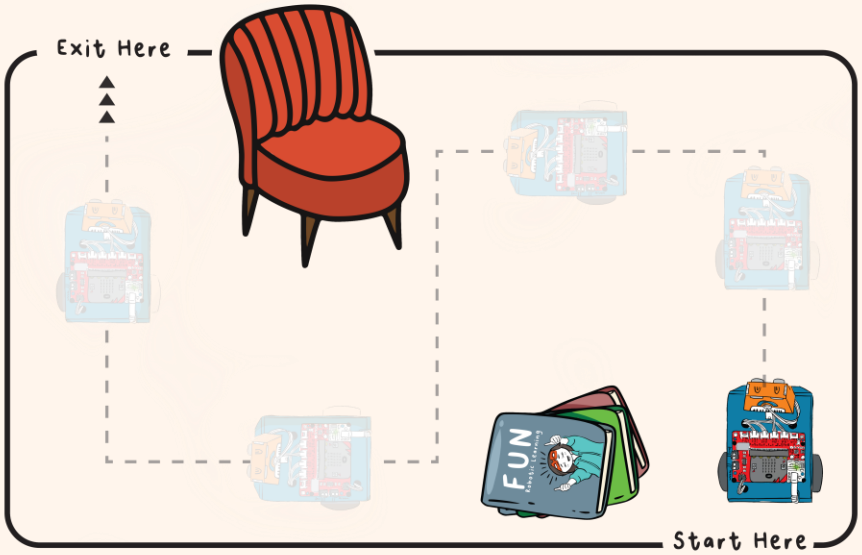
4 按下 按钮 A，按钮 B 和 按钮 A+B。观察 ZOOM:BIT 智能小车的反应。

当 按钮 A 被按下时	向右转， 维持 500 毫秒 (ms)
当 按钮 B 被按下时	向左转， 维持 500 毫秒 (ms)
当 按钮 A 和 B 同时被按下时	向前行驶， 维持 1 秒 (second)



为你准备了一个有趣的挑战！

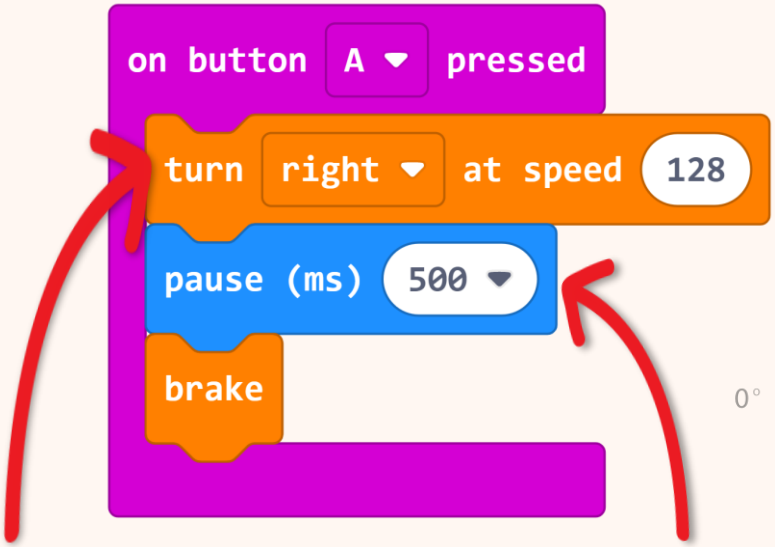
寻找一个空阔的地方，以及在 ZOOM:BIT 的行驶道路随机摆设一些障碍物，
例如椅子、书本和纸盒。邀请你的兄弟姐妹和朋友们来挑战吧！
让他们手动操控及引导 ZOOM:BIT 以寻找到正确的路线。

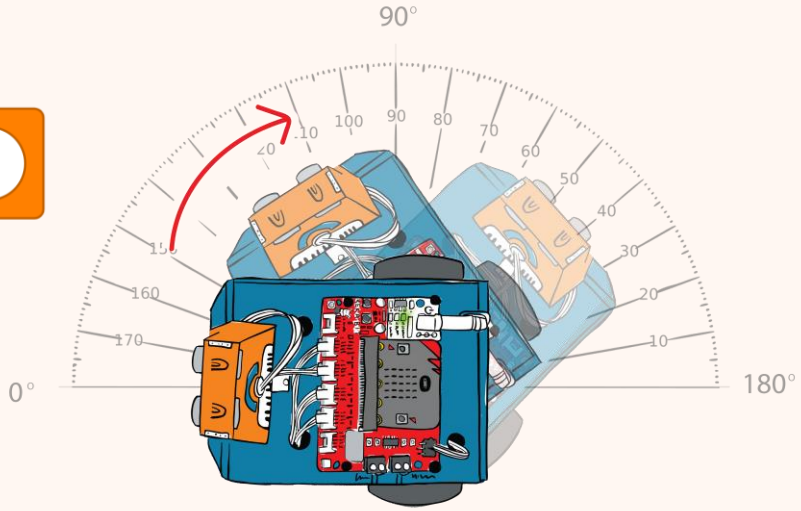


- ❑ 同时按下 按钮 A 和 B 向前行驶
- ❑ 按下按钮 A 转右
- ❑ 按下按钮 B 转左

使用最短时间 (或最少移动次数) 来操控及引导 ZOOM:BIT
走出障碍物地图的挑战者将获得胜利！

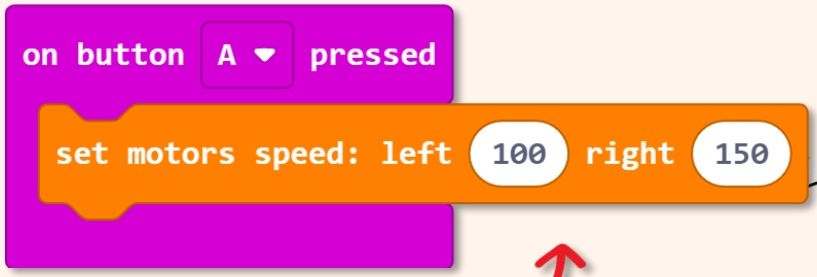
你是否知道？ 你可以更改速度 speed 和 延迟 (时长) delay，以控制旋转的角度。你可以尝试使用各种数值来完成以下的表格。



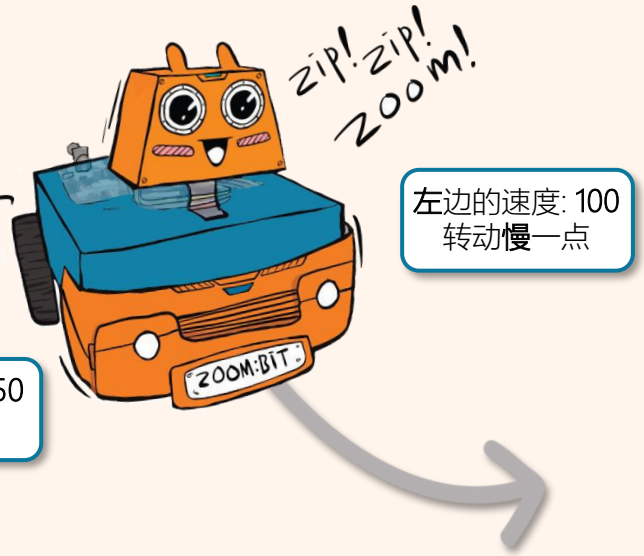


速度	延迟 (时长)	角度
128	500	
	250	60
255	500	

探索更多编码块



使用这个编码块来调整轮子旋转的速度。



你是否知道？

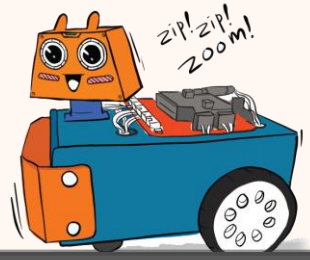
如果两边的轮子旋转的速度不相同，ZOOM:BIT 将**转向** (steer) 旋转比较慢的轮子方向。举个例子，ZOOM:BIT 将向前行驶但是转向偏左的方向，这是因为左边轮子的转速比较慢，而右边比较快。

如果 ZOOM:BIT 的**左轮**速度为 150，**右轮**速度为 200；你是否能够预测 ZOOM:BIT 移动的方向呢？测试并认证你是否正确吧！

你是否知道？

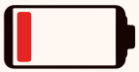
其实，马达的规格与实际性能之间也有不可避免的细微差别。即使提供相同的**电压** (voltage)，看似相同的马达也可能拥有略微不同的旋转速度。换句话说，即使你将 ZOOM:BIT 编程为直线行驶 (左右轮速度相同)，但在一段时间后，ZOOM:BIT 仍然可能偏向右，或偏向左行驶。

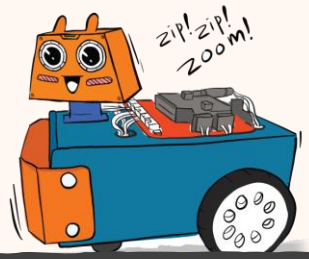
ZOOM:BIT 移动的 准确性 accuracy 和 一致性 consistency 也会被 **电池电量 battery level** 和 **表面状况 condition of surface** 所影响。当电池电量比较低，或行驶的表面太软和不平坦时，将会导致 ZOOM:BIT 行驶得比较慢。



坚硬光滑的表面，例如大理石地板

行驶比较快





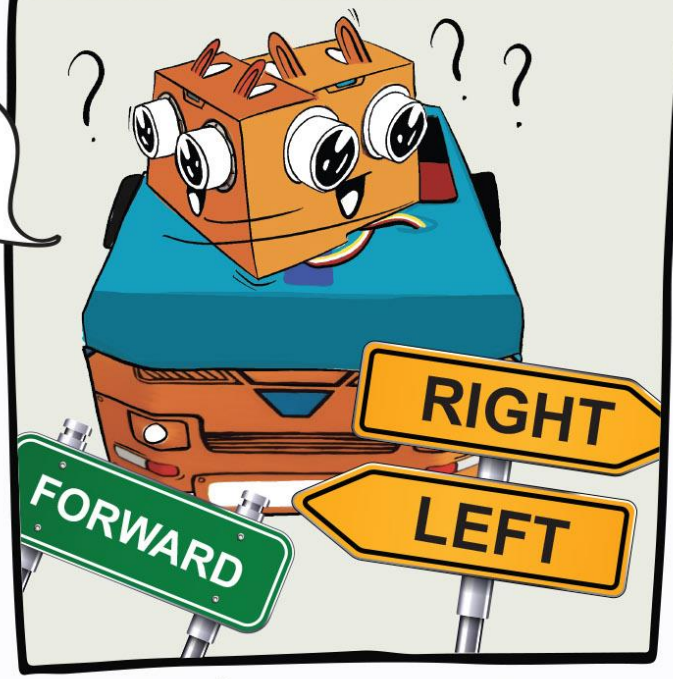
柔软不平坦的表面，例如地毯

行驶比较慢

CHAPTER 5 第五章

https://link.cytron.io
/zoombit-chapter-5

LET'S START!

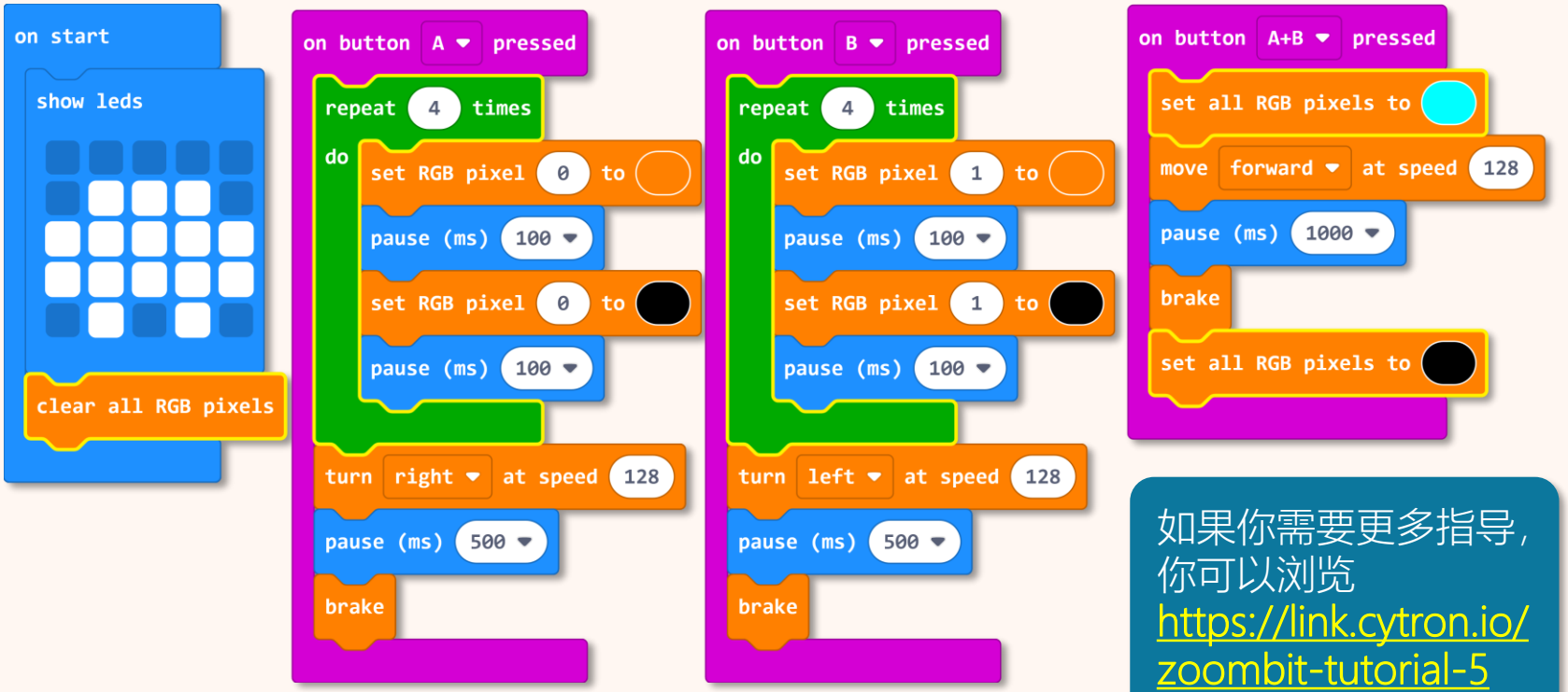


Left? Right? Please Signal
Where You're Going~

左还是右？请记得发出信号～

REKA:BIT 板上拥有两个标记了 “0” 和 “1” 的 RGB LED 灯。你可以使用 REKA:BIT 分类抽屉中的编码块来续写让 LED 灯以不同的颜色点亮的程序。

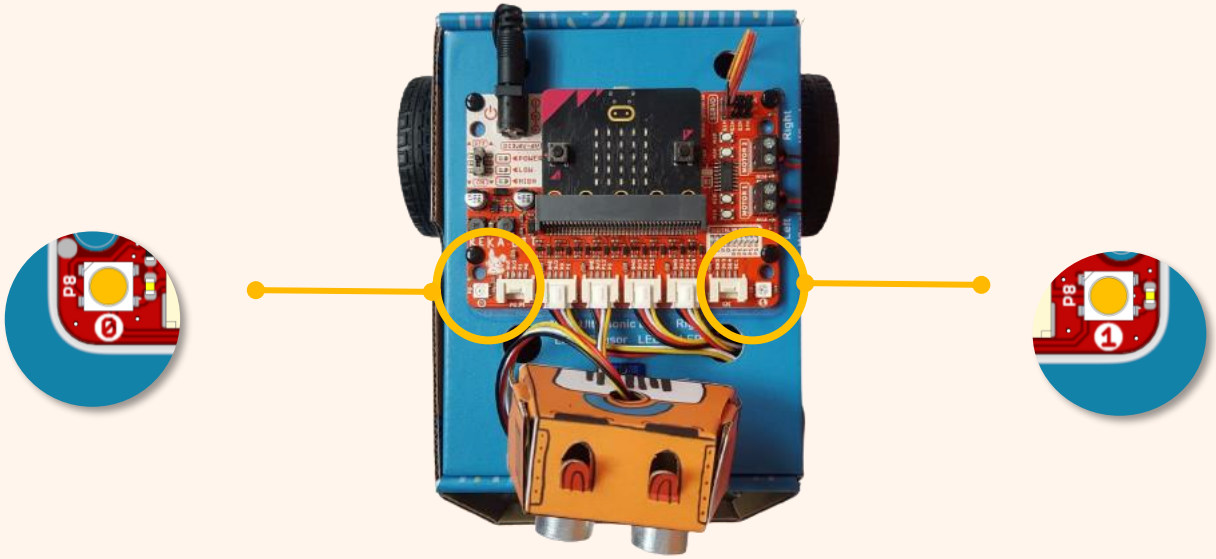
1 延续前一课的程序，从 [Basic]【基本】，[Loops]【重复】及 [REKA:BIT] 分类抽屉中获取图中突出显示的编码块，添加至你的程序里。



如果你需要更多指导，你可以浏览 https://link.cytron.io/zoombit-tutorial-5 获取更多分步指南。

2 把程序下载到 ZOOM:BIT，然后打开电源键。

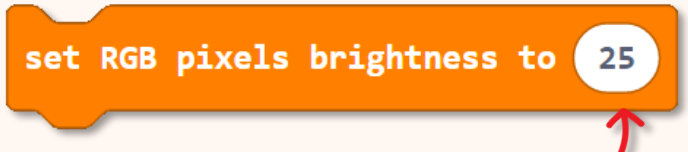
3 按下按钮 A，按钮 B 和 按钮 A+B。观察 REKA:BIT 上 RGB LED 灯的反应。



你有没有发现到 ZOOM:BIT 在向右转前，右边的 “0”号 RGB LED 灯会先闪烁?
而 ZOOM:BIT 在向左转前，左边的 “1”号 RGB LED 灯会先闪烁?
然后当 ZOOM:BIT 要往前行驶时，两边的 RGB LED 灯则会亮起蓝色呢?

探索更多编码块

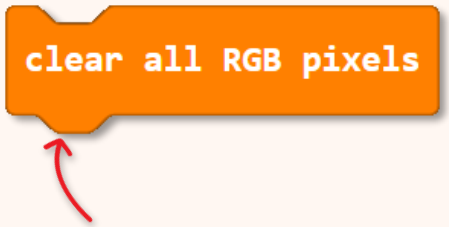
设 RGB 像素 pixel(s) 到所选的颜色。若要更换颜色，请点击最右边的椭圆形状并在调色板里选择你所要的颜色。



调整 RGB 像素的亮度。
亮度值在 0 至 255 (最亮) 之间。

黑色 = 关闭 RGB LED 灯。

若你要的颜色**没有**出现在调色板上，你可以使用这个编码块来自己**拟定颜色**。



关闭所有的 RGB 像素。

为你准备了一个有趣的挑战！

请问你可以续写一个程序来让 ZOOM:BIT 的 RGB LED 灯以警车的应急灯方式亮起吗？
为了有更好的效果，你可以让它发出警笛声吗？

* 在警笛的部分，你可以选择 中音 C *(middle C)* 和 中音 F# *(middle F#)* 来重复播放。

* 如果你正在使用没有内置扬声器的 micro:bit 第一代，你可以跳过警笛的部分。

CHAPTER 6 第六章

https://link.cytron.io/zoombit-chapter-6

LET'S START!

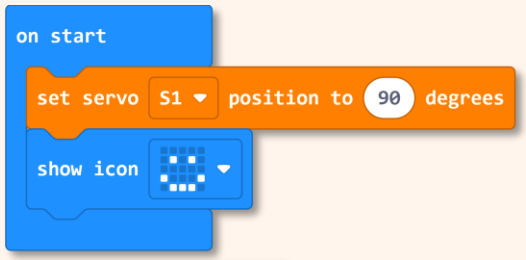
一起跳舞吧！

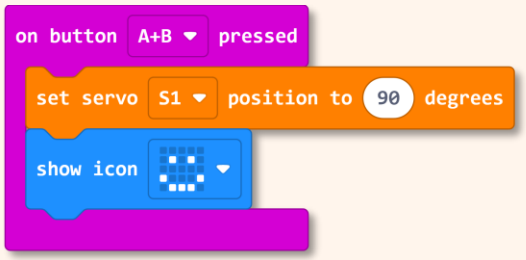
ZOOM:BIT 的头部是连接至一个可以旋转 180°度 的 **伺服电机 (servo motor)** 。
换句话说，你可以编写 ZOOM:BIT 并控制伺服电机的角度来让它向前看，
转左和转右，或转向任何指定的角度。让我们来试一试吧！

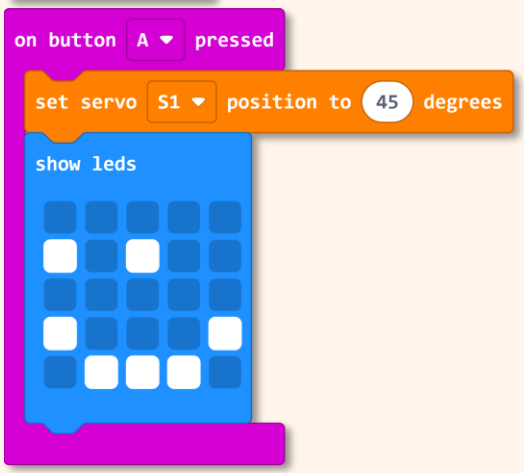
1 在你的 MakeCode Editor 中创建一个新项目。 然后添加 ZOOM:BIT 的扩展。
(你可以参考第 44-45 页)

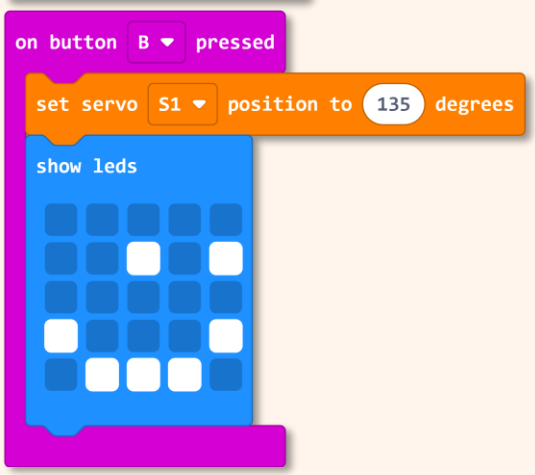
2 续写以下程序。你可以到以下分类抽屉寻找你所需要的编码块。

Basic | REKA:BIT | Input





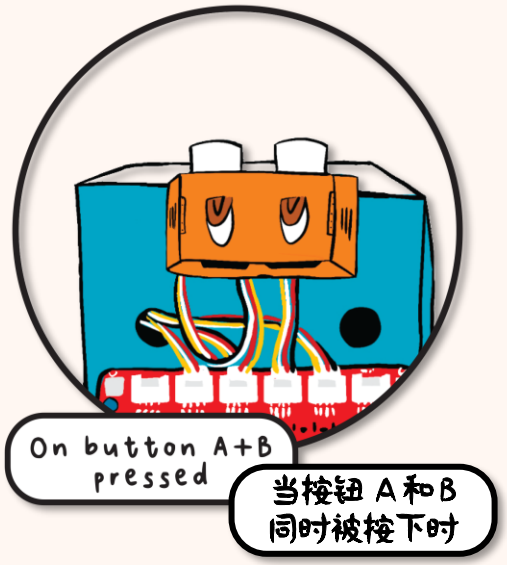


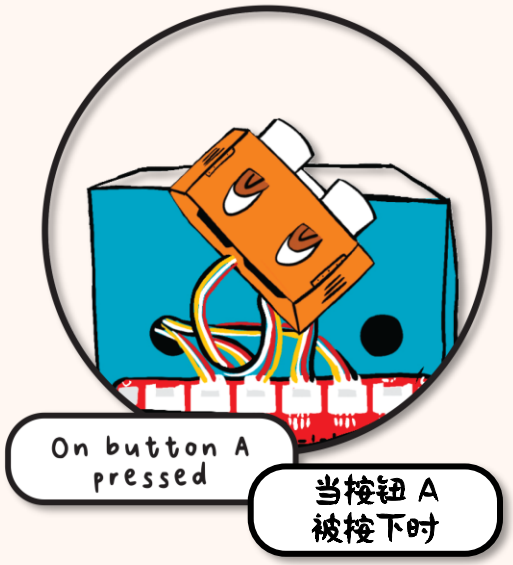


如果你需要更多指导，请浏览 https://link.cytron.io/zoombit-tutorial-6 以获取更多分步指南。

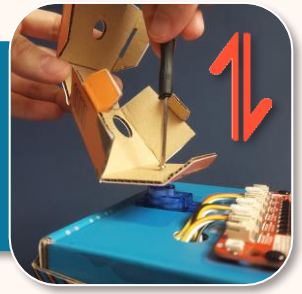
3 把程序下载到 ZOOM:BIT ，然后打开电源键。

4 按下 按钮 A，按钮 B 和 按钮 A+B；观察 ZOOM:BIT 头部的方向以确保头部所面向的方向是正确的。





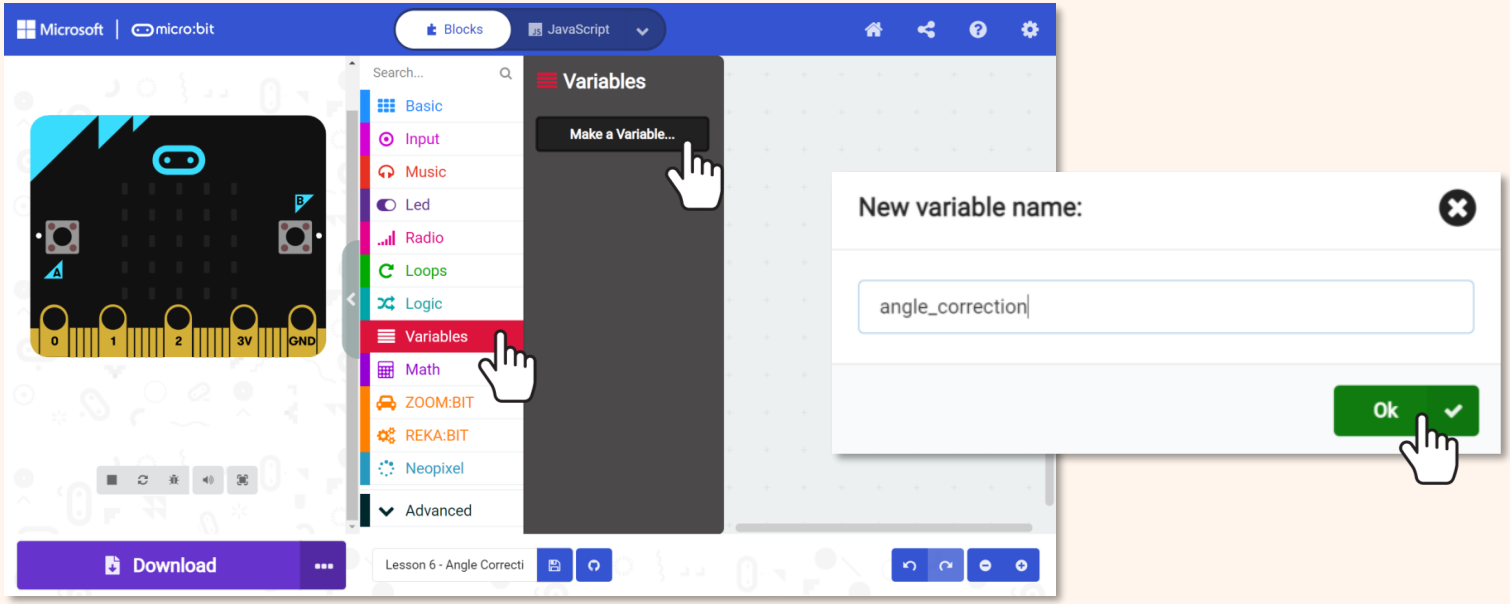
当你同时按下了按钮 A 和 B 时，请问你的 ZOOM:BIT 头是否有向前看呢？如果它不是正对齐的话，那你就得拧开 ZOOM:BIT 头部的螺丝，矫正了之后再将它锁回伺服电机架子上。



如果你在手动矫正 ZOOM:BIT 头部后，仍然发现它的头部还是有点向左或右偏移，你可以在程序里进行进一步的调整。根据以下的步骤来确定你的 ZOOM:BIT 头部的矫正角度。

1 在你的 MakeCode Editor 中创建一个新项目。 然后添加 ZOOM:BIT 的扩展 。 *(你可以参考第 44-45 页)*

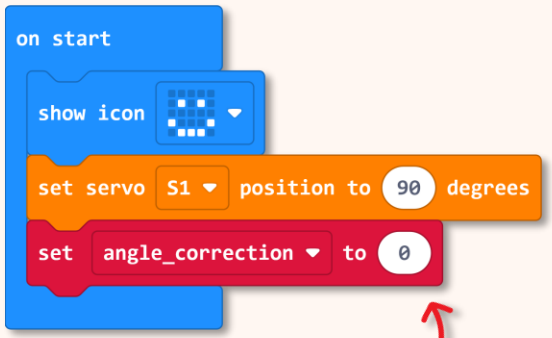
2 点击 [Variables]【变量】然后选择 [Make a Variable...]【设计变量】。命名你的变量（例: “angle_correction”）然后点击 [Ok] 键。



3

续写以下代码。你可以到以下分类抽屉寻找你所需要的编码块。

Basic | REKA:BIT | Variables | Input | Logic | Math



当开机时，设变量 [angle_correction] 为 0。

为了得到更多指导，请浏览 https://link.cytron.io/zoom bit-tutorial-6a 以获取更多分步指南。

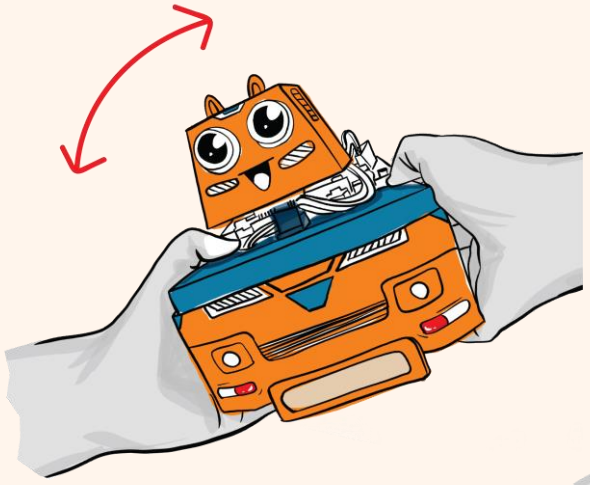
若 ZOOM:BIT 向**右**倾斜，变量 [angle_correction] 将会改变 1，ZOOM:BIT 头部会**向右**旋转 1°度。

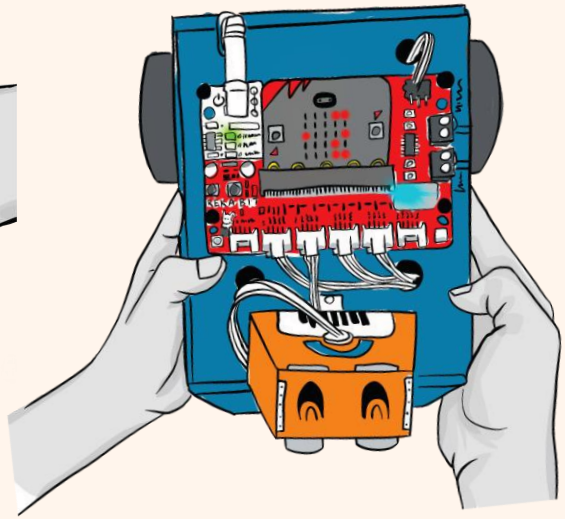
若 ZOOM:BIT 向**左**倾斜，变量 [angle_correction] 会改变 -1，ZOOM:BIT 头部会**向左**旋转 1°度。

为了阅读变量 [angle_correction] 的数值，把 ZOOM:BIT 抬高，以便 micro:bit 的**徽标 logo 朝上** (ZOOM:BIT 的头部向下看) 。

4 把程序下载到 ZOOM:BIT，然后打开电源键。把 ZOOM:BIT 向左或右倾斜以让它的头部转向指定的方向。

5 当你满意了 ZOOM:BIT 头部向前看的角度，把 ZOOM:BIT 提起，micro:bit 的 logo 朝上（ZOOM:BIT 的头向下），然后读取 “angle_correction” 的数值。





把 [angle_correction] 的数值记录在这里。

现在你知道了 ZOOM:BIT [angle_correction] 的变量数值，你可以在未来的项目里利用这个数值，以确保头部始终保持在你要的角度。

填写之前已经记录好的
[angle_correction] 数值

这里是使用了变量
[angle_correction]
的一个示例代码。

当电源键打开后，
ZOOM:BIT 会向左看，
然后向右看，
最后向前看。

为你准备了一个有趣的挑战！

请问你可以续写一个程序来让 ZOOM:BIT 跳舞吗?
让我们一起发挥创意，让 ZOOM:BIT 尽情地旋转和摇摆吧 ~

当你使用 micro:bit 第二代时，你可以从 [Input]【输入】 类别里使用 [On Loud Sound]【(大声) 声音】 编码块来触发 ZOOM:BIT 开始跳舞；你也可以添加 [Music]【音乐】编码块来让整个舞蹈表演更加轻快有趣。

CHAPTER 7 第七章

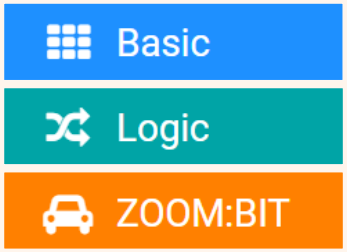
https://link.cytron.io/zoombit-chapter-7

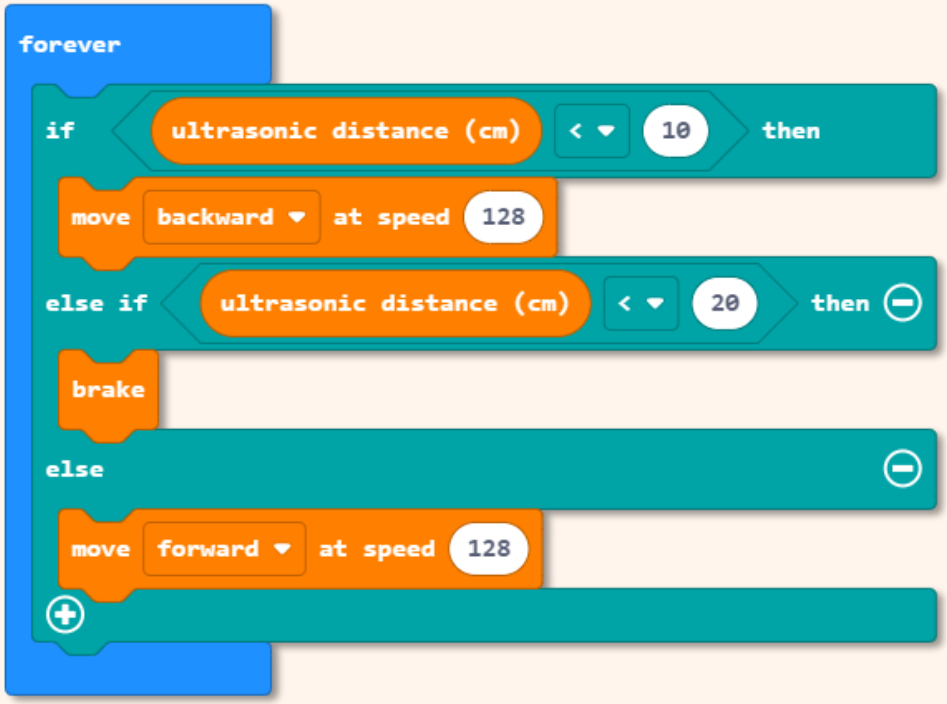
前方有障碍物！

现在 ZOOM:BIT 是可以移动了，让我们一起教它避免撞到障碍物吧！

1 在你的 MakeCode Editor 中创建一个新项目。 然后添加 ZOOM:BIT 的扩展。
(你可以参考第 44-45 页)

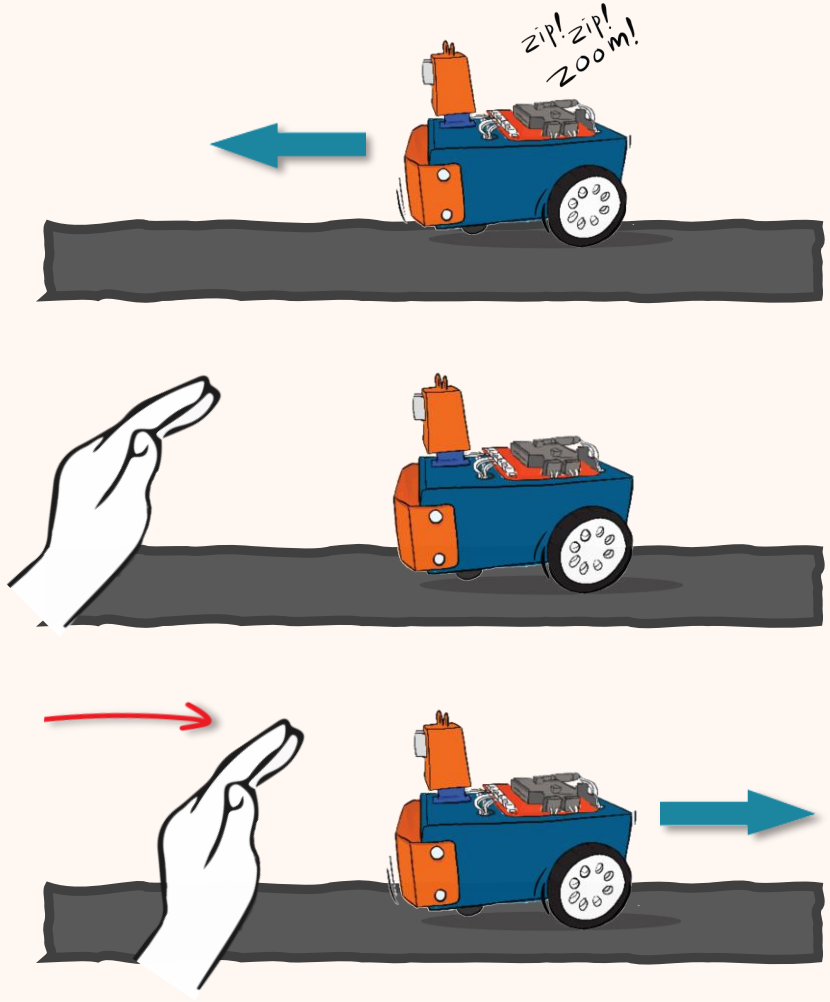
2 续写以下程序。你可以到以下分类抽屉寻找你所需的编码块。





如果你需要更多指导，你可以浏览 https://link.cytron.io/zoombit-tutorial-7 获取更多分步指南。

3 把程序下载到 ZOOM:BIT ， 然后打开电源键。



如果前方没有障碍物，ZOOM:BIT 将会不断向前行驶。

试着把你的手放在 ZOOM:BIT 的前方。请问智能小车是不是在离你的手 10 厘米 (cm) 时停下来呢?

慢慢把你的手向 ZOOM:BIT 靠近。观察智能小车离你的手少过 10 厘米 (cm) 时的反应。

让我们编程让 ZOOM:BIT 在没有前行时，如果 按钮 A 被按下，ZOOM:BIT 会向右转；而当 按钮 B 被按下时，ZOOM:BIT 会向左转。
例：在离障碍物 10 厘米 (cm) 时停下。

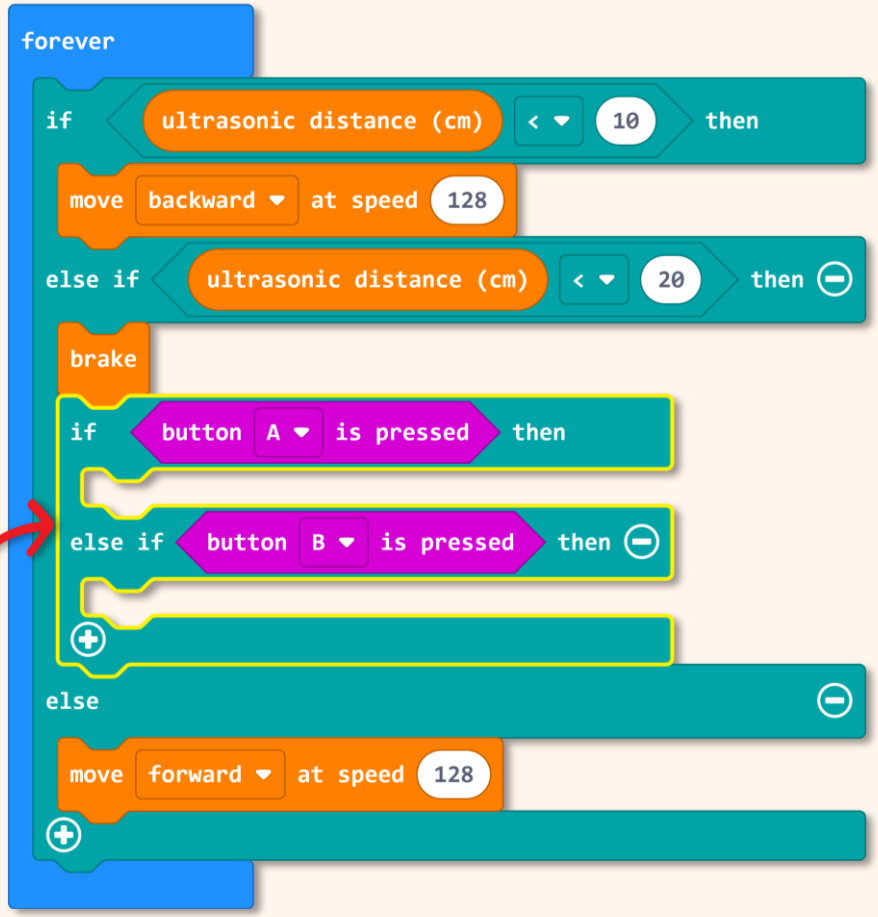
4 在 [brake] 编码块之后添加以下被突出显示的编码块。

你可以到以下分类抽屉寻找你所需要的编码块。

Input

Logic

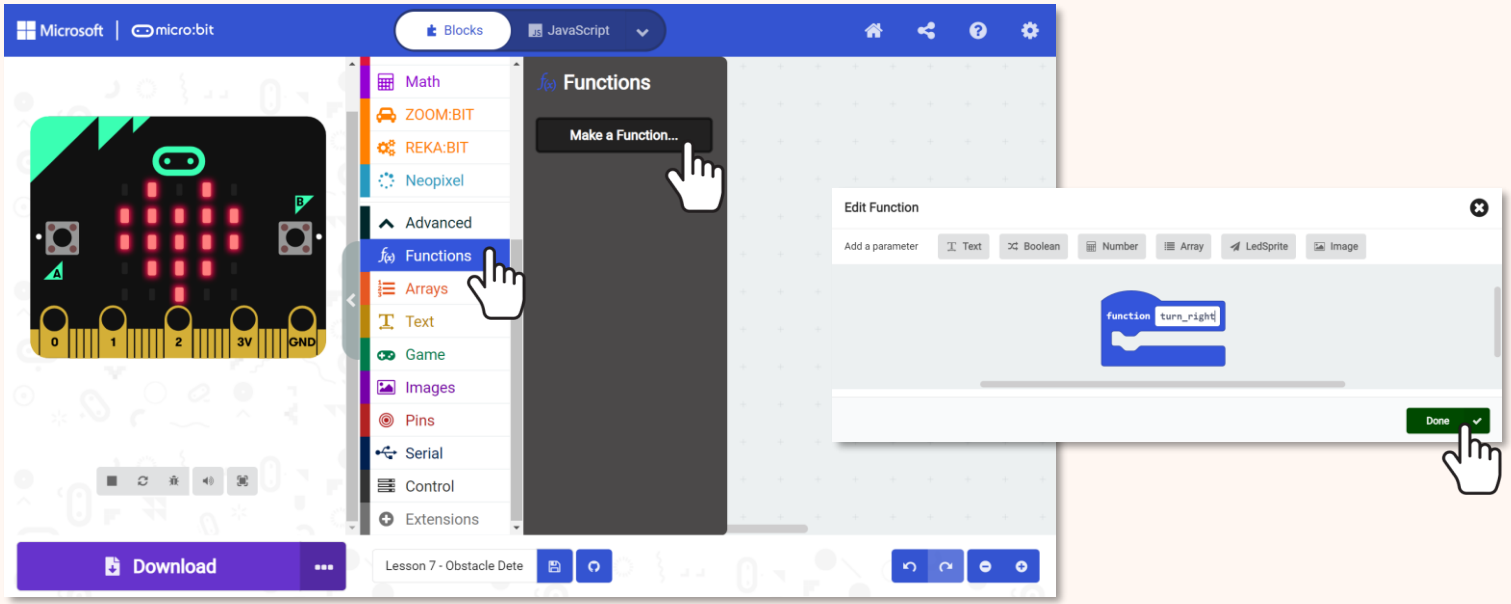
这是叫做
“nested if condition”
“嵌套如果条件”



你是否知道？

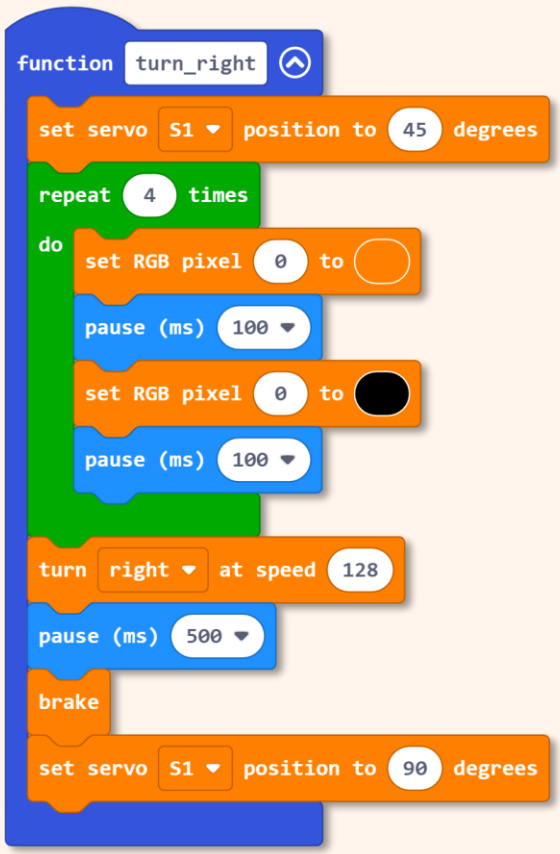
我们可以将执行特定任务的编码块创建成 **函数** (functions)。在创建了一个 function 后，你并不用一直续写同样的程序编码块，而是可以在程序中重复使用 **函数** 的功能 。此外，专业的程序员也会使用 **函数** 的功能来让他们的程序更简单易懂。

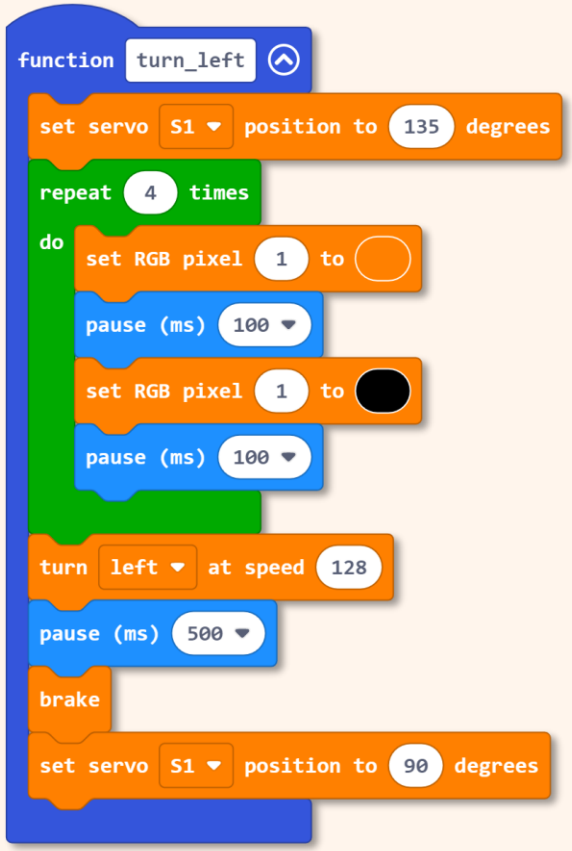
5 点击 [Advanced]【高级】然后选择 [Functions]【函数】分类抽屉。点击 [Make a Function]【创建一个函数】，然后将 (doSomething) 更改成 **"turn_right"**，然后点击 [Done]【完成】。一个 [function turn_right]【函数 turn_right】的编码块将会添加至你的编程工作区。



6 重复以上步骤，创建一个新的函数编码块 (functions)，并将它改名为 “turn_left”。

7 接着，利用以下的编码块来完成 [function turn_right]【函数 turn_right】和 [function turn_left]【函数 turn_left】的程序.



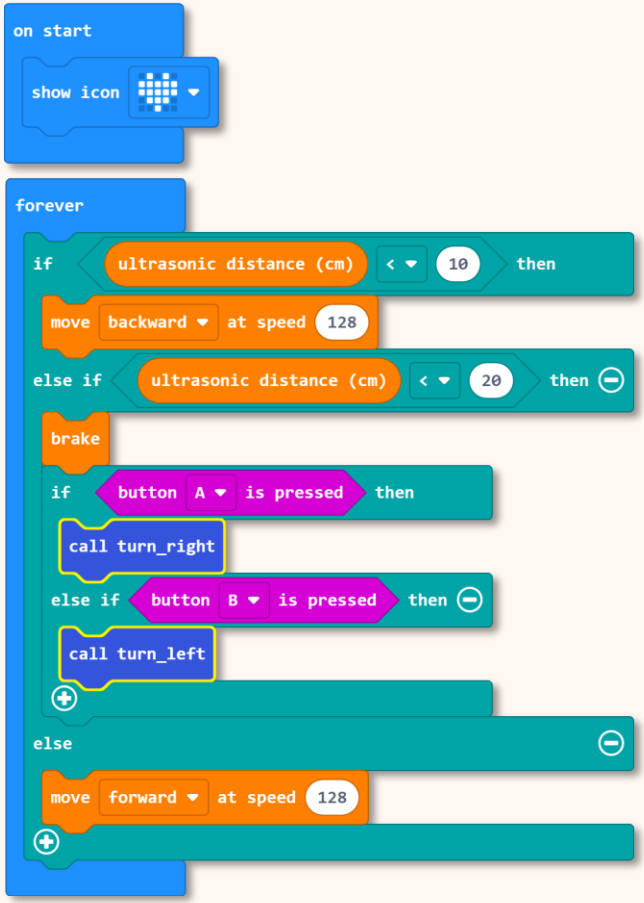


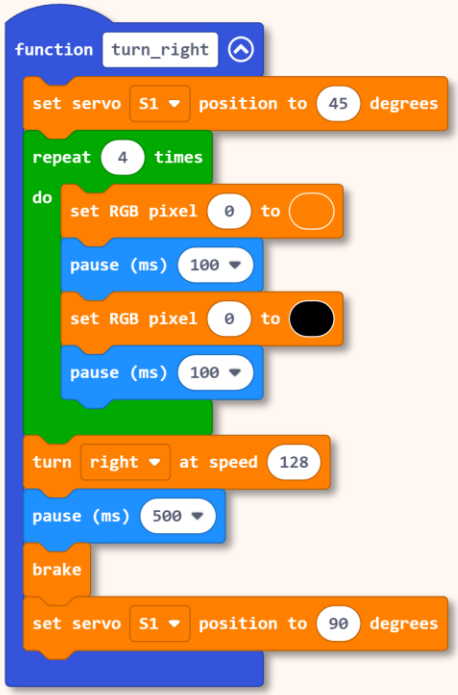
在完成函数 functions 编码块后，你可以点击 ⌃ 按钮来**折叠**所有的编码块。

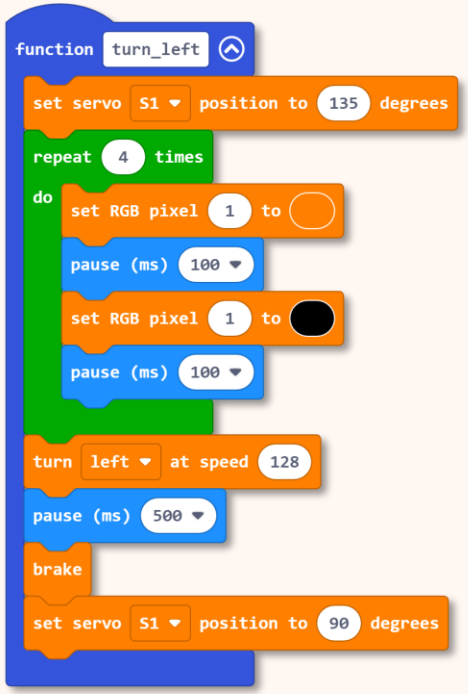
如果你需要审查或更改你的程序，点击 ⌄ 按钮，**展开**函数块。

8

最后，点击 [Functions]【函数】分类抽屉，然后添加 [call turn_right]【调用 turn_right】和 [call turn_left]【调用 turn_left】函数编码块至你的程序。这里是一个完整的程序：

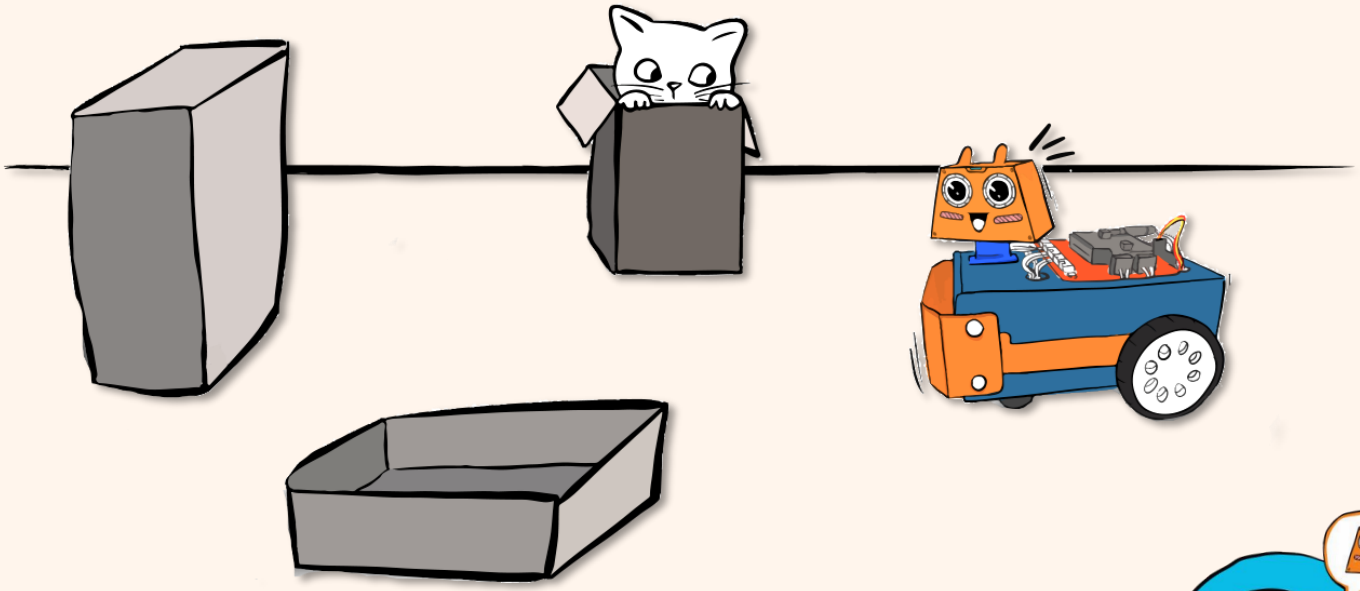






Yeah！现在 ZOOM:BIT 可以在不会撞到障碍物的情况下，自由地在你的房间里行驶了。当 ZOOM:BIT 的道路被障碍物挡着时，你可以按下按钮 A (向右转) 或 按钮 B (向左转) 来引导 ZOOM:BIT 越过所有障碍物。

9 把程序下载到 ZOOM:BIT ， 然后打开电源键。



请问你觉得你可以自行修改此程序来让 ZOOM:BIT 在没有你的帮助下自动避开所有的障碍物吗？一起试试看吧~

为你准备了一个有趣的挑战！

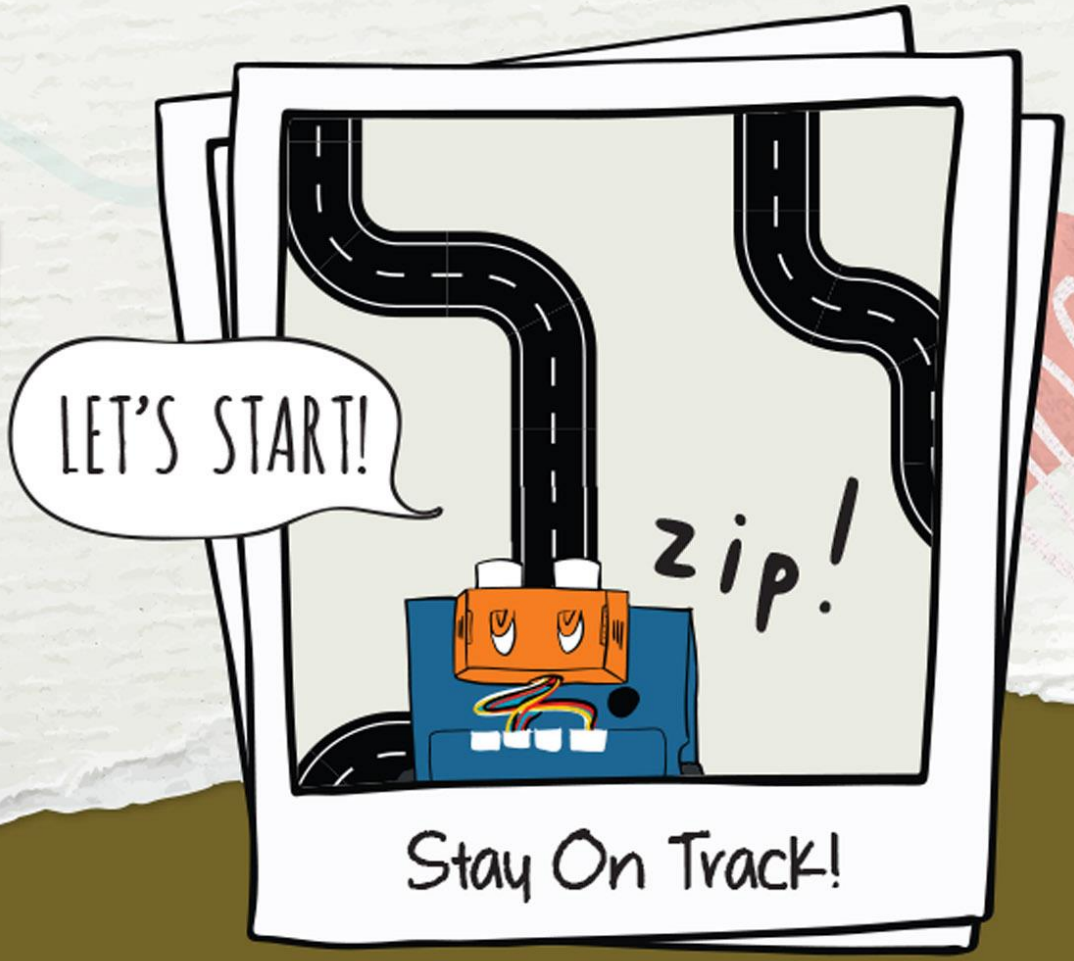
把 ZOOM:BIT 改造成一个超声波钢琴吧！
请续写一个程序来让 ZOOM:BIT 根据超声波传感器的读数，播放不同的声调。

当一个物体被放置在离 ZOOM:BIT						
< 5	< 10	< 15	< 20	< 25	< 30	< 35
___ 厘米(cm)，播放音调						
C	D	E	F	G	A	B
维持 ½ 节拍和展示音调的字母。						

来让 ZOOM:BIT 唱首歌吧！将你的手心放在 ZOOM:BIT 的前方并向前移或向后移来让它播放不同的声调。
如果你不清楚如何运作，你可以扫描左边的二维码来观看演示短片。

CHAPTER 8 第八章

https://link.cytron.io/zoombit-chapter-8



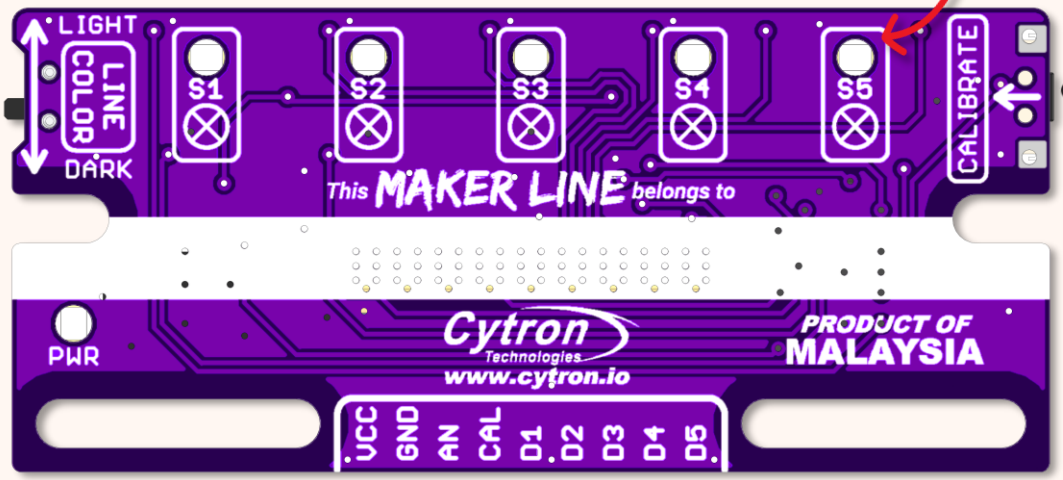
保持在跑道上行驶

你是否知道？

你知道 ZOOM:BIT 可以被编程以跟着线行驶吗？
ZOOM:BIT 可以轻易完成这项任务因为它配备了 Maker Line 巡线传感器。
这个传感器的功能是它能够探测线路 (黑或白) 与对比颜色的背景。

当线条被相应的 IR 传感器探测到时，那个相应的 IR 传感器 (S1-S5) 里的 LED 指示器将会亮起。

当所供应的轨道颜色是黑色的 (与白色背景比较)，滑动线条颜色的开关去 "DARK"。



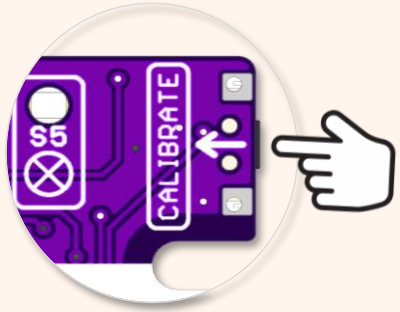
按压这个按钮来校准 calibrate Maker Line 传感器

Maker Line 巡线传感器 – 顶视图

在你开始编写 ZOOM:BIT 的程序前，请跟着以下的步骤来校准 Maker Line 巡线传感器。校准步骤只需进行一次即可；除非传感器的高度，线的颜色或者背景颜色被更换。

1. 展开所提供的 ZOOM 跑道。放置 ZOOM:BIT 至跑道上，并打开 ZOOM:BIT 的电源键。

2. 按住写着 (CALIBRATE)【校准】按钮**不动**直至所有 (5个) 的 LED 灯亮起；当所有的 LED 灯开始闪烁后，方可松开此按钮。



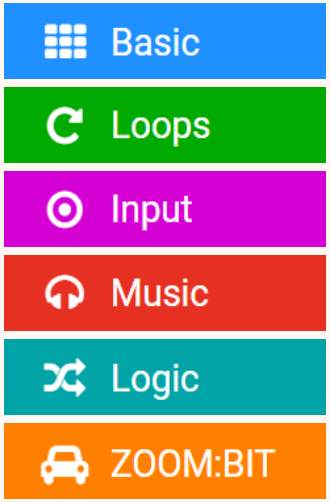
3. 将 ZOOM:BIT 左右移动，让 Maker Line 巡线传感器在黑色线条上经过。重复移动几次，以确保所有的传感器都感应到黑色线。

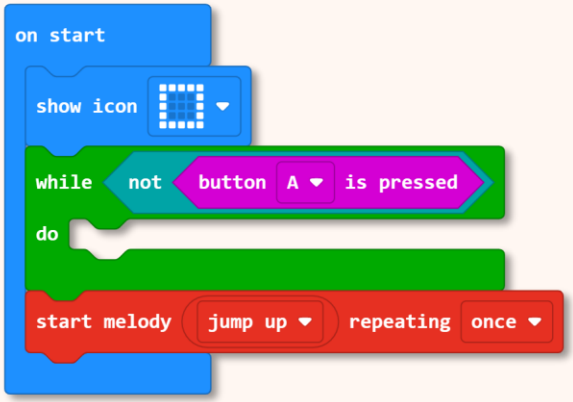
4. 重新按压 (CALIBRATE)【校准】按钮以结束校准模式。

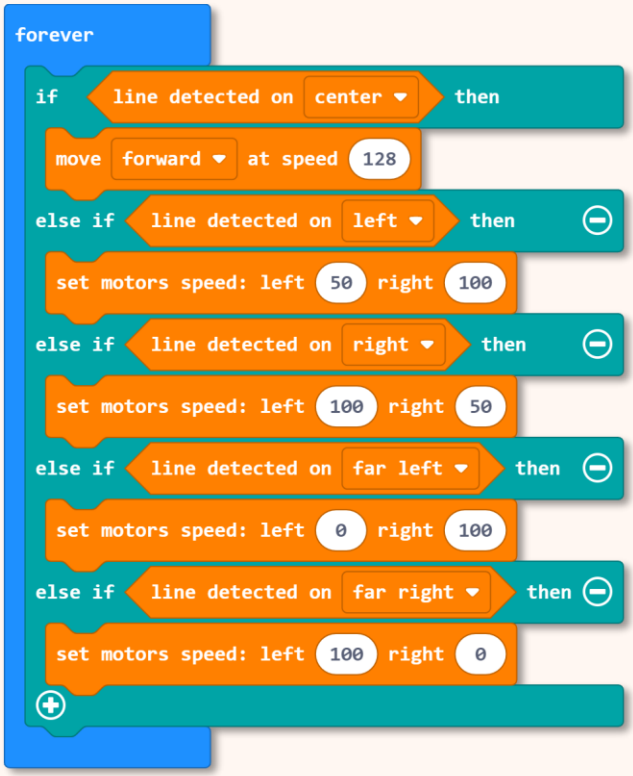
如果校准成功，你会看见跑灯的效果；接着你的 Maker Line 已经可以使用了。如果你还不清楚应该做什么或者如何操作，你可以扫描左边的二维码来观看一个演示视频。

1. 在你的 MakeCode Editor 中创建一个新项目。 然后添加 ZOOM:BIT 的扩展。 *(你可以参考第 44-45 页)*

2. 续写以下程序来引导 ZOOM:BIT 跟着跑道行驶。你可以到以下分类抽屉寻找你所需要的编码块。

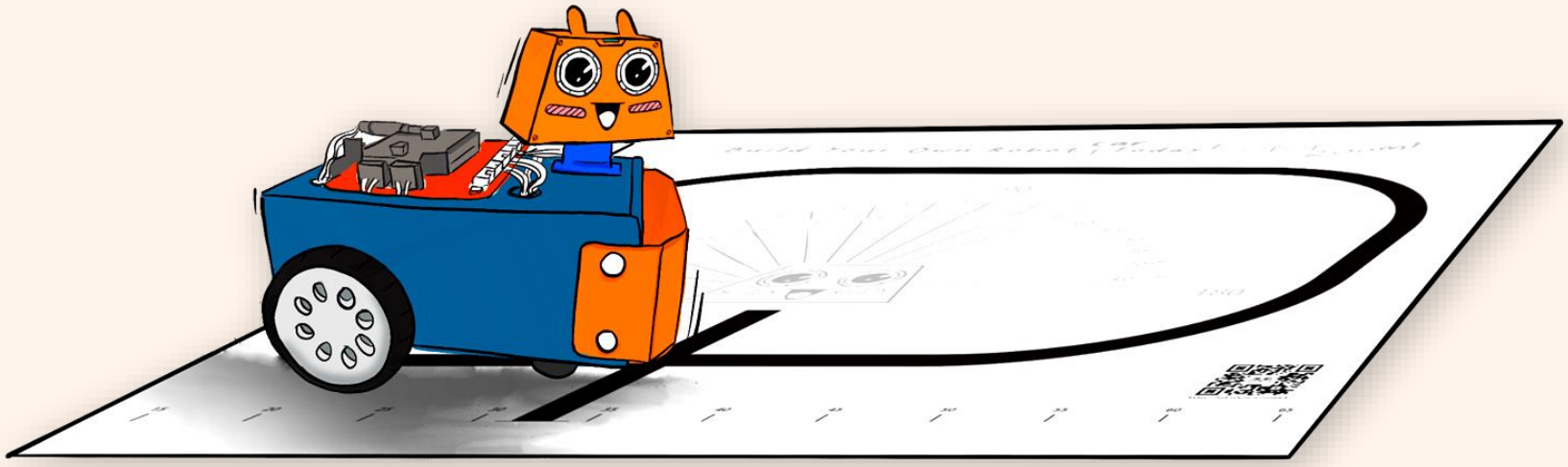






如果你需要更多指导，你可以浏览 https://link.cytron.io/zoombit-tutorial-8 获取更多分步指南。

3 把程序下载到 ZOOM:BIT ， 然后打开电源键。将它放置在黑色跑道，接着按下 按钮 A 。



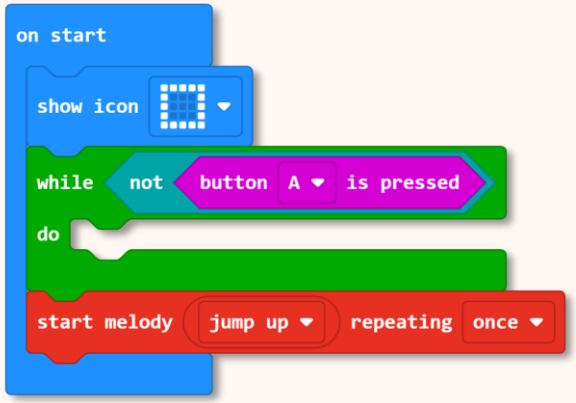
在你按下 按钮 A 后，细看 ZOOM:BIT 如何不断地巡着跑道行驶。请问你能理解这个程序的操作吗？

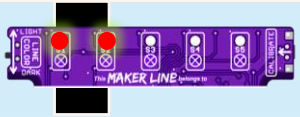
在程序一开始时，显示 □ 图像。

在 按钮 A 还没被按下前，不做任何动作。

若 按钮 A 被按下，退出条件循环 while loop。
播放 “jump up” 旋律一次。

不断检查 Maker Line 读数，并做出相对的回应。



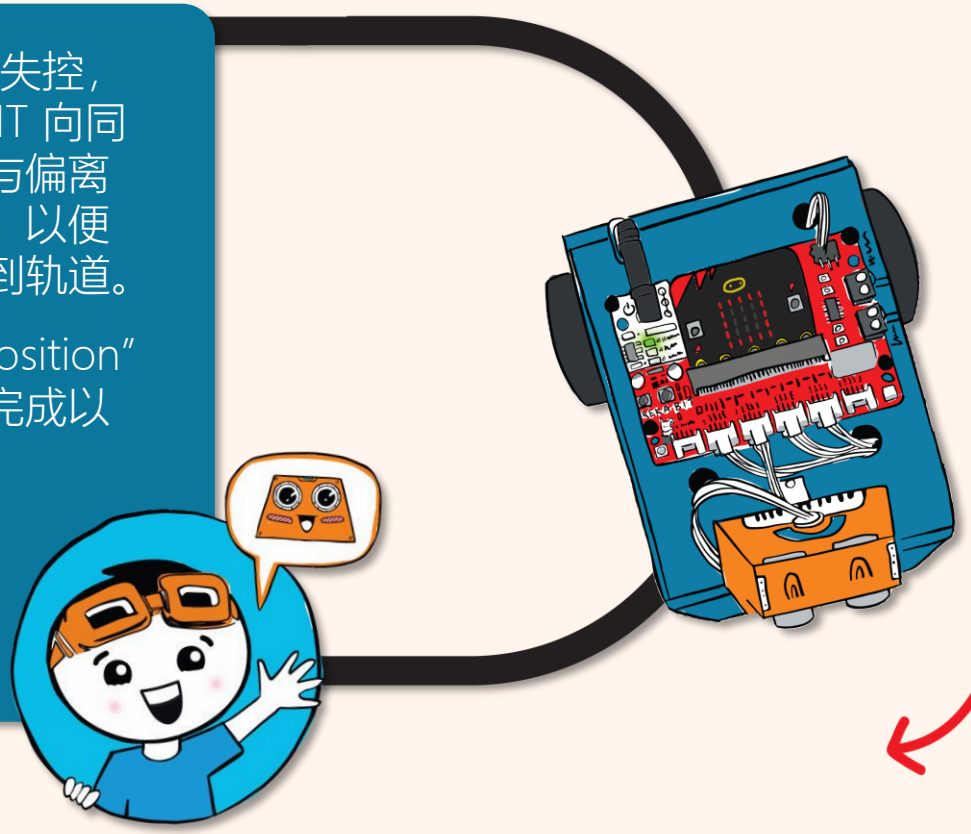
条件	黑线被探测的位置	返回轨道的动作
	中间	往前行驶
	左边	轻微往左转
	右边	轻微往右转
	最左边	往左转
	最右边	往右转

请问你的 ZOOM:BIT 有时候会偏离轨道吗？尤其是当它面临转弯的时候。当 ZOOM:BIT 在转角处时，它的 Maker Line 巡线传感器可能会暂时偏离黑色轨道（如下所示）。这种情况会使 ZOOM:BIT 不知所措，因为在前面的代码，我们没有告诉 ZOOM:BIT 当黑线没有被探测到时，应该如何做出反应。

为了避免 ZOOM:BIT 失控，我们要教导 ZOOM:BIT 向同一个方向进行转弯（与偏离轨道前的方向一致），以便它能寻找，并成功回到轨道。

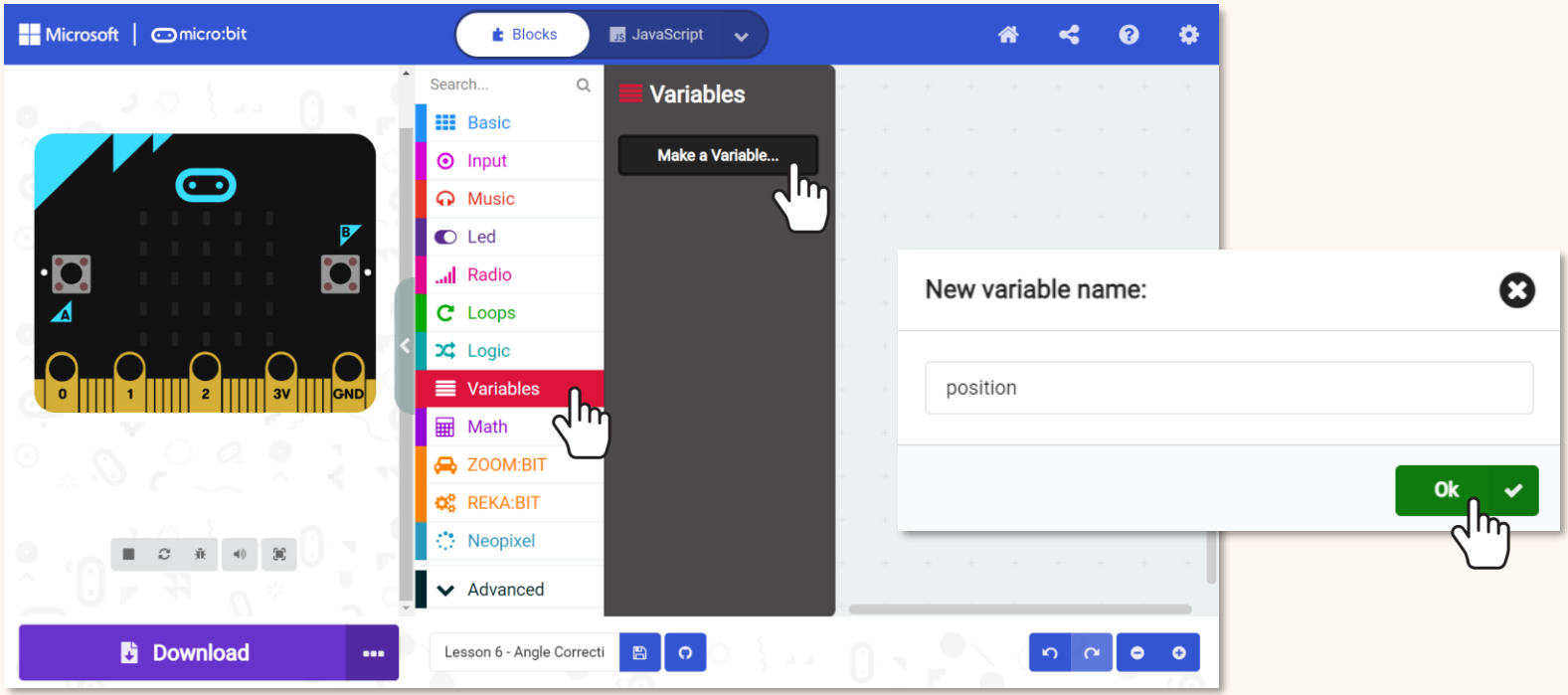
我们可以添加一个 “position” 变量到我们的程序来完成以上的目的。

请翻开下一页来学习如何更好地改善我们的程序。



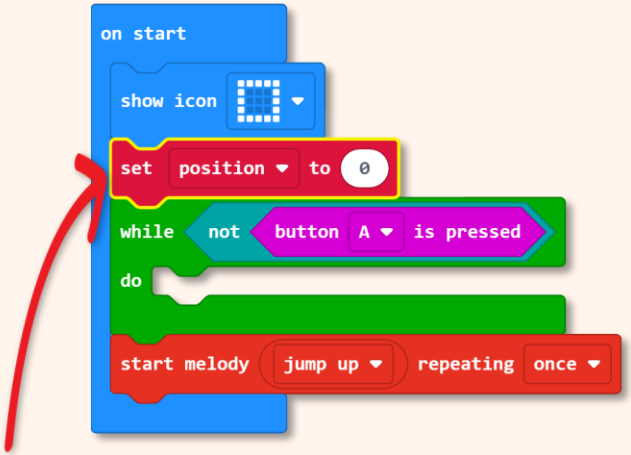
4

点击 [Variables]【变量】分类抽屉，然后选择 [Make a Variable...]【设计变量】。命名你所创建的变量（例："position"），然后点击 [OK]【确定】按钮。

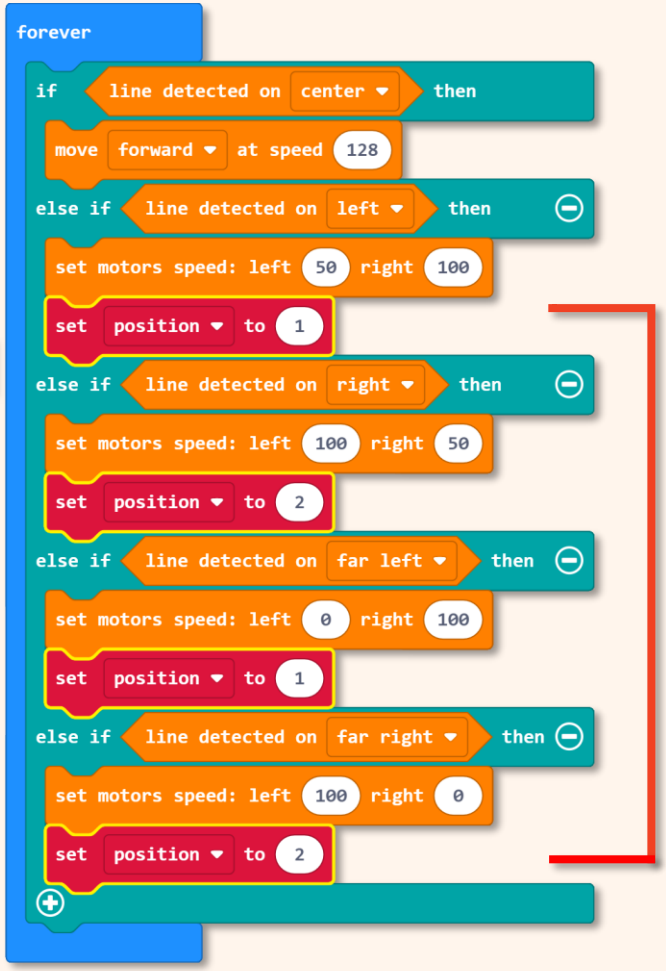


5

从 [Variables] 【变量】分类抽屉里选择 [set (position) to (__)] 【将 (position) 设为 (__)】编码块，添加至你的程序。



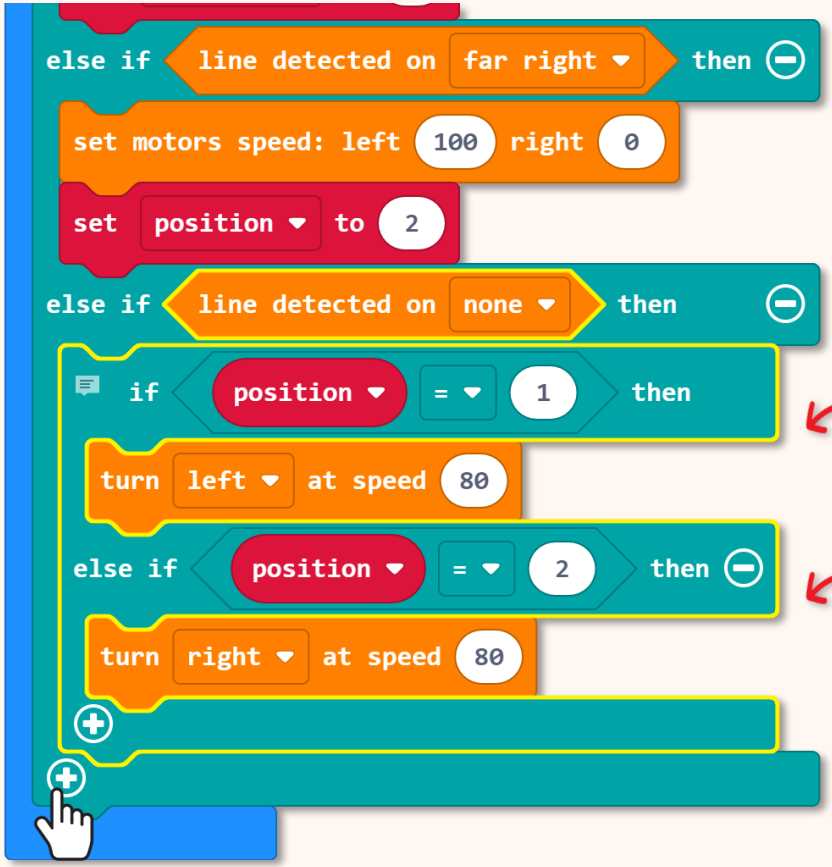
当 ZOOM:BIT 的电源键被打开时，变量 [position] 将会被设置至 0。



当线条在**左边**被探测到时，设置变量 [position] 至 “1”；
而当线条在**右边**被探测到时，设置变量 [position] 至 “2”.

6

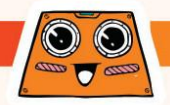
点击 ⊕ 按钮，添加一个 **"else-if 否则-如果"** 的条件。
然后再添加以下的编码块至你的程序。



这是一个新的条件。
"当没有线条被探测到时"。

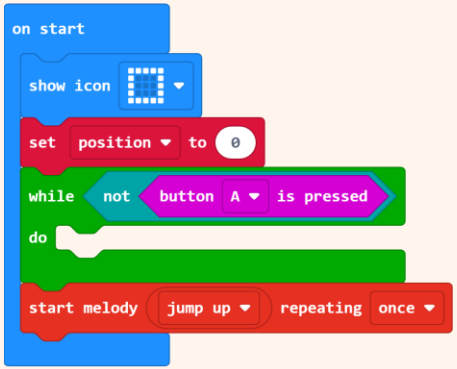
这是一个嵌套如果条件。
"检查 [position] 变量，
如果为 1，就转**左**"。

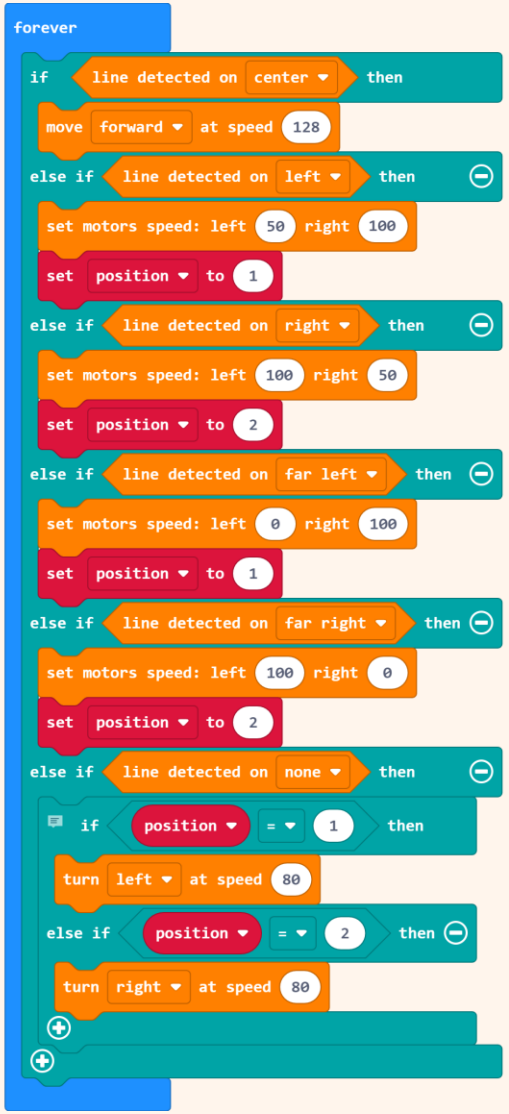
否则 " [position] 变量
如果为 2，就转**右**"。



7

把完整的程序下载到 ZOOM:BIT ，然后打开电源键。
将它放置在跑道上。
接着，按下 按钮 A 。



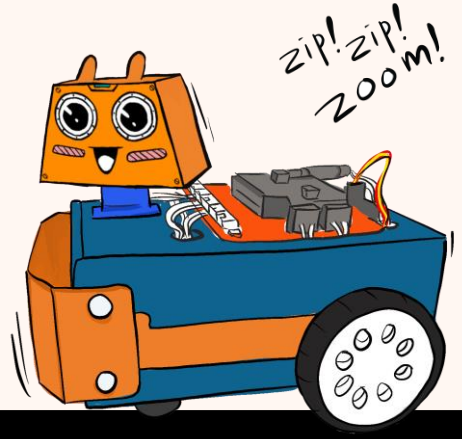


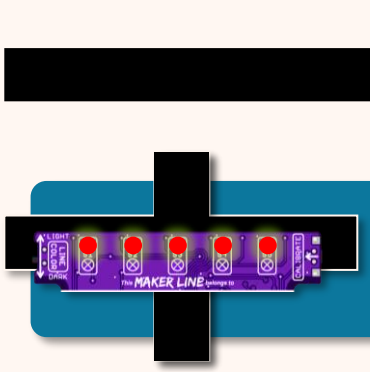
让我们进行测试。试着把 ZOOM:BIT 推出轨道（没有接触到黑线）。你是否发现 ZOOM:BIT 会自己调整位置并重新回到跑道上，而没有直接失控呢?

为你准备了一个有趣的挑战！

你可以续写一个程序让 ZOOM:BIT 来完成以下的任务吗?

- 当 按钮 A 被按下时，ZOOM:BIT 会沿着跑道行驶。
- 每当 ZOOM:BIT 跑过终点线时，播放一个 1/2 节拍的音调
- 显示 ZOOM:BIT 所行驶的轨道圈数 lap(s)，和
- 让 ZOOM:BIT 在行驶了三圈整 (3) 后停下来。





提示：我们可以利用 **[line detected on (all)]** 编码块来让 ZOOM:BIT 知道它已经越过终点线。

你是否知道？

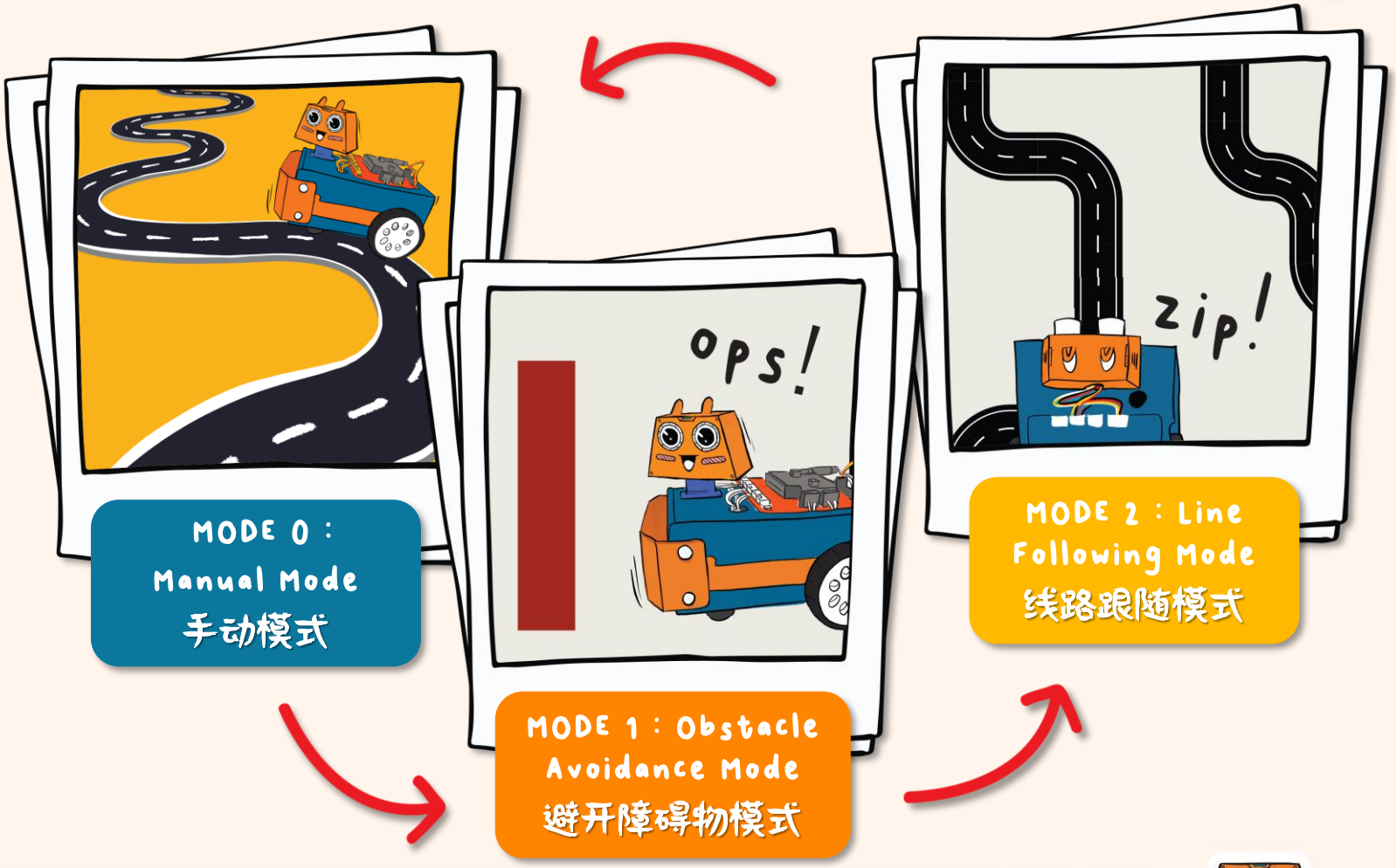
除了使用已提供的轨道，你也可以发挥创意并使用黑色的电工胶带 black vinyl electrical tape 来制造属于你自己的跑道。你可以轻易的在五金店找到这种材料。当你们设计属于自己的赛道时，祝你们玩得开心！

CHAPTER 9 第九章

https://link.cytron.io/zoombit-chapter-9

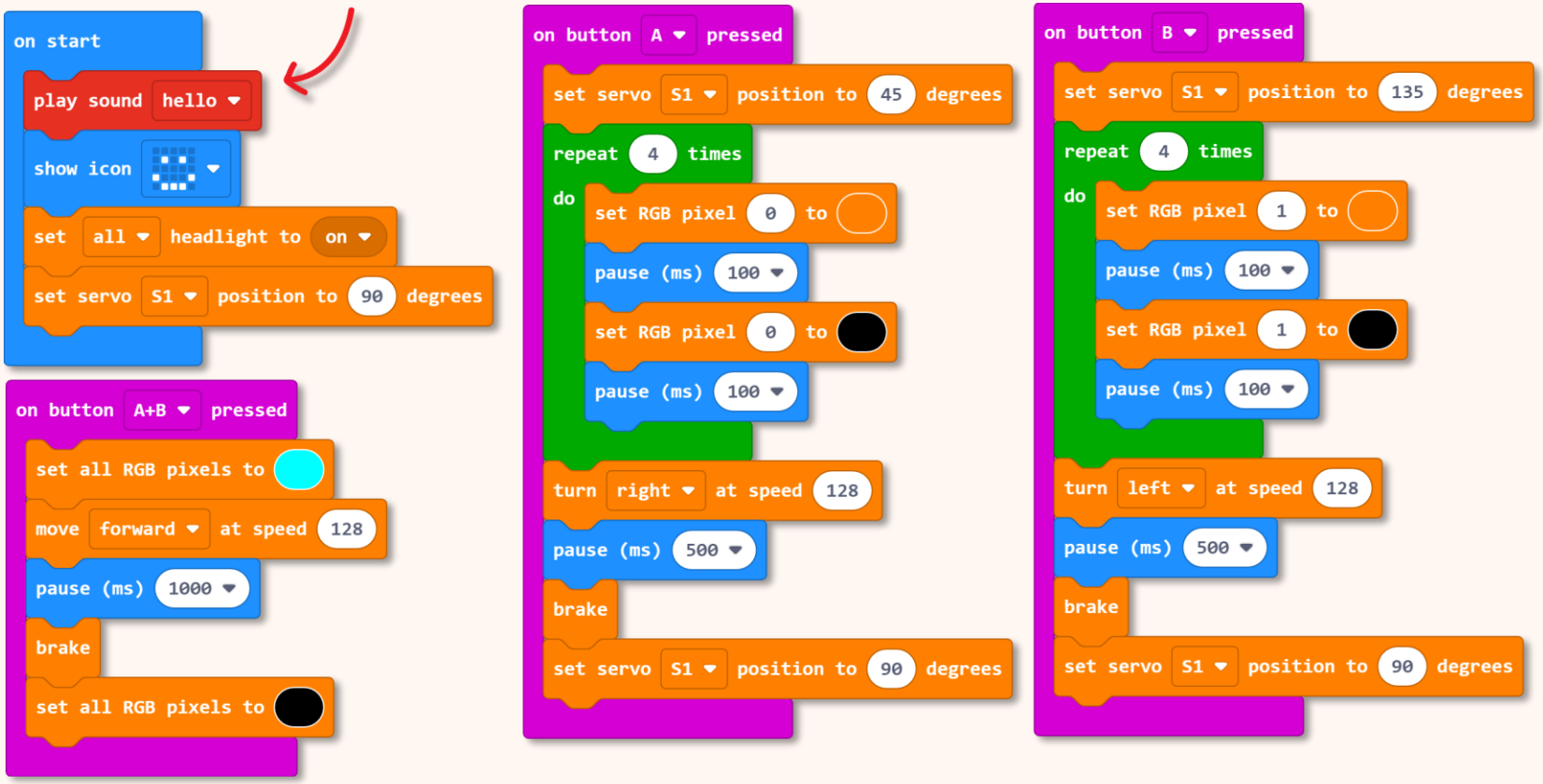
把所有都合在一起。你能做到的！

我们已经一起教导了 ZOOM:BIT 许多技巧，而它也已经全部学会了。
现在让我们一起训练 ZOOM:BIT 能随时随地地切换模式吧！



1 在你的 MakeCode Editor 中创建一个新项目，然后添加 ZOOM:BIT 的扩展 。*(你可以参考第 44-45 页)*。接着，编写以下 **手动模式 manual mode** 的程序。

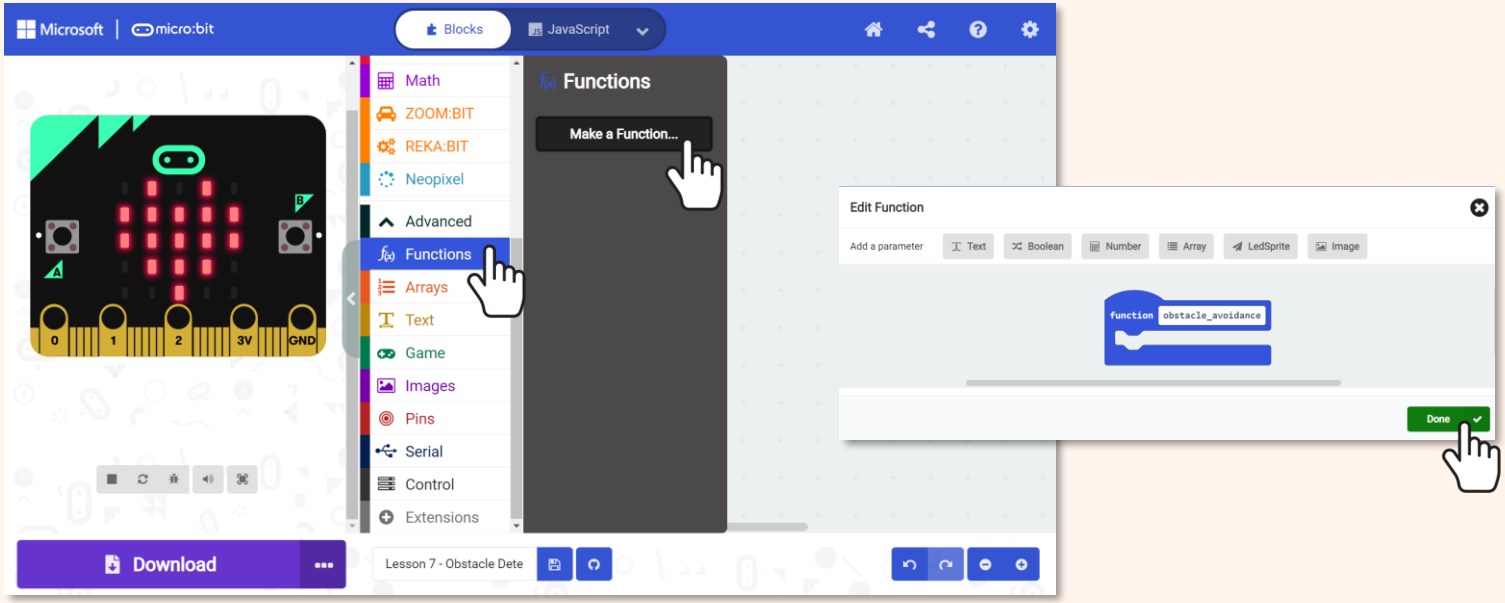
若你使用的是 micro:bit 第一代，
请忽略以下的音乐编码块。



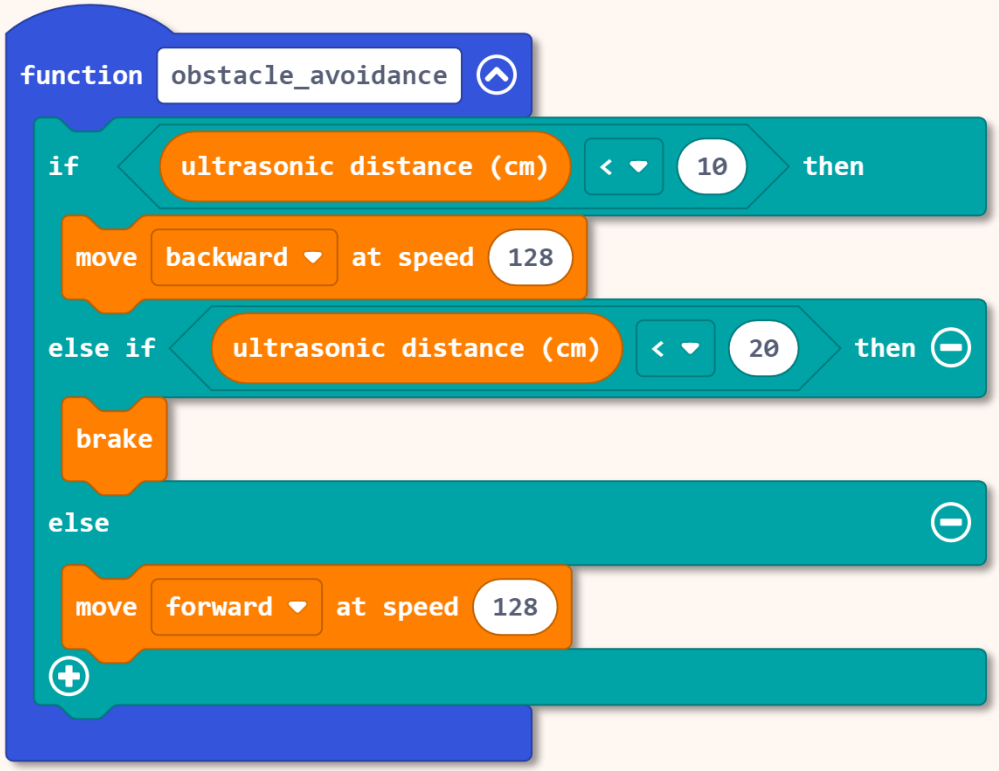
现在让我们添加其他的模式。为了完成它，我们需要使用 **函数** functions。

2

点击 [Advanced]【**高级**】 然后选择 [Functions]【**函数**】分类抽屉。点击 [Make a Function]【**创建一个函数**】，将 doSomething 更改为 **'obstacle_avoidance'**，然后点击 [Done] 按钮。一个 [function obstacle_avoidance] 【**函数** obstacle_avoidance】函数编码块将会添加至你的编程工作区。



3 添加其他编码块至你的 [obstacle_avoidance] 函数编码块下，完成 **避除障碍物模式 obstacle avoidance mode**。



在完成函数 functions 编码块后，你可以点击 按钮来**折叠**所有的编码块。

如果你需要审查或更改你的程序，点击 按钮，**展开**函数块。

你有没有发现到这个程序与第七章里的程序相似呢？但是在这里使用的是 [function obstacle_avoidance] 函数编码块而不是 [forever] 编码块。

4 重复第二个步骤 (Step 2) 来创建另一个函数 (function) 给予 **线路跟随模式line following mode。**

5 如右图所示，添加编码块至 [function line_following]【**函数** line_following】函数编码块下。

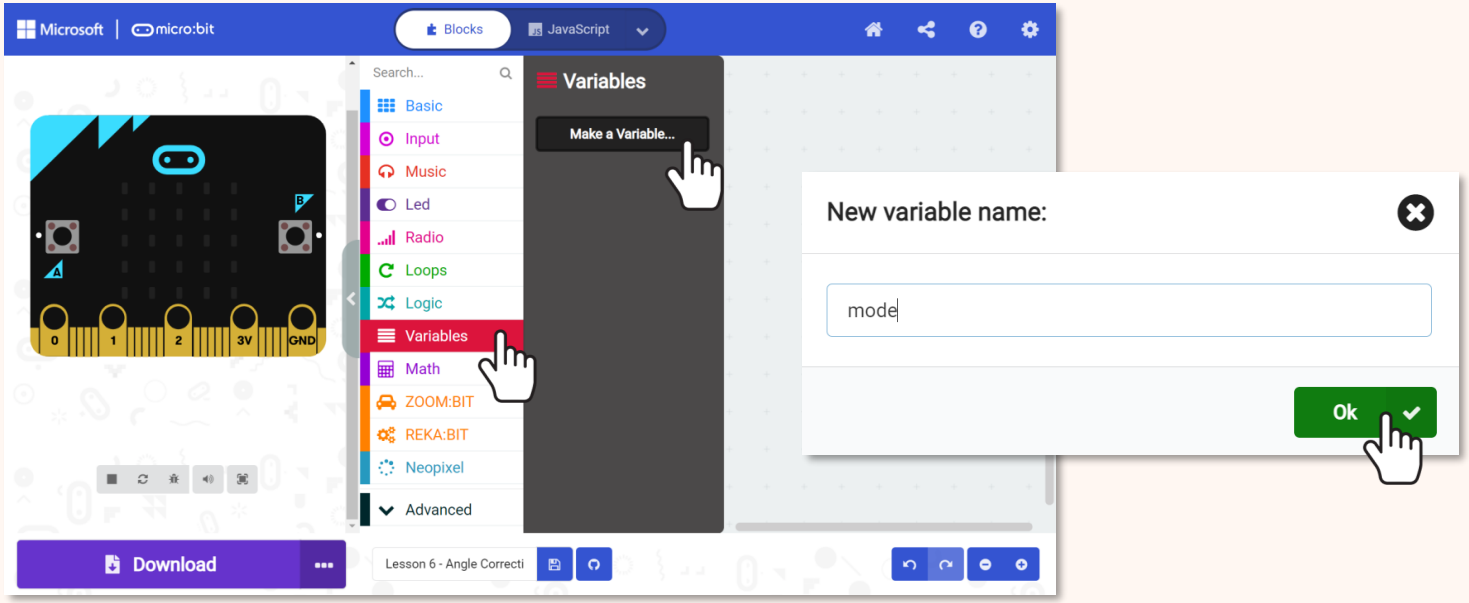
这个程序与 Chapter 8 里的程序相似，但是在这里使用的是 [function line_following] 函数编码块而不是 [forever] 编码块。

***注意事项**：你需要创建一个新的变量 [position] 。

如果你需要更多指导，你可以浏览 https://link.cytron.io/zoombit-tutorial-9 获取更多分步指南。

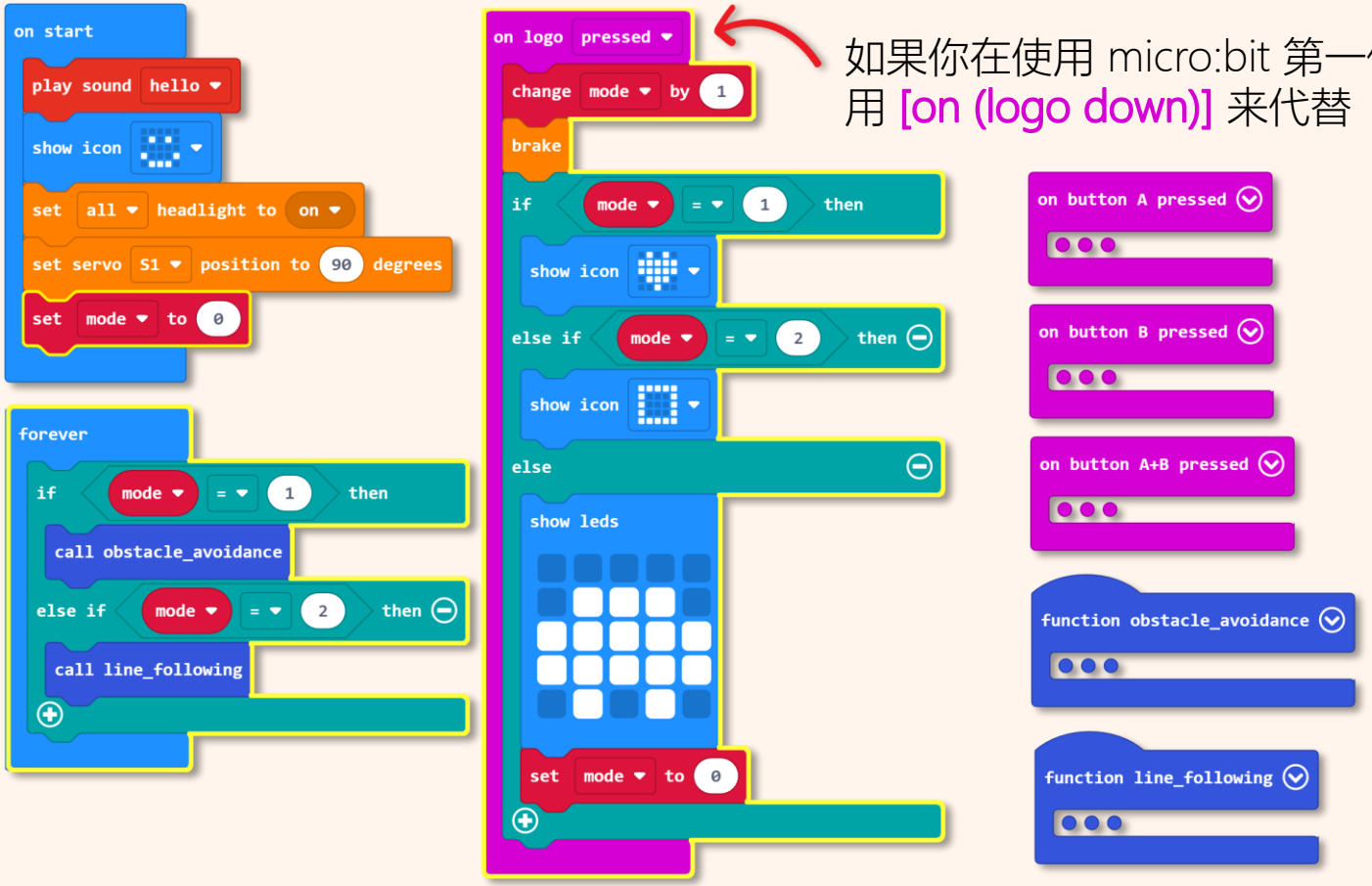
接着，我们将会添加“modes 模式”至我们的程序，方便我们切换模式，ZOOM:BIT 会自动执行我们所选择的特定模式里的相应任务。

6 点击 [Variables]【变量】然后选择 [Make a Variable...]【设计变量】。命名你的变量（例:“mode”），然后点击 [Ok]【确定】键。



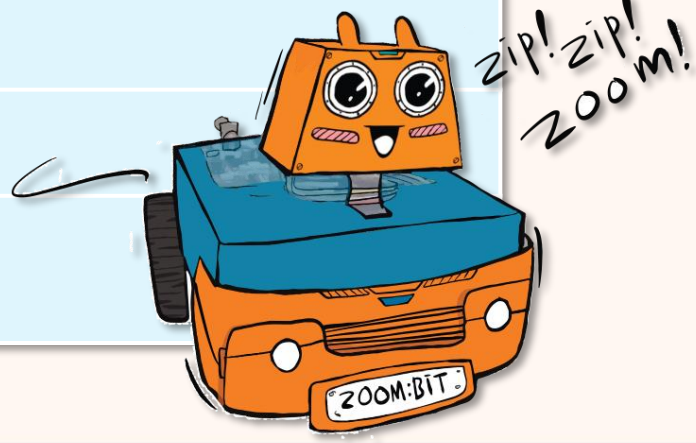
7 添加以下代码至你的程序。你可以到以下分类抽屉寻找你所需要的编码块。

Basic | Variables | Logic | Functions | Input | ZOOM:BIT



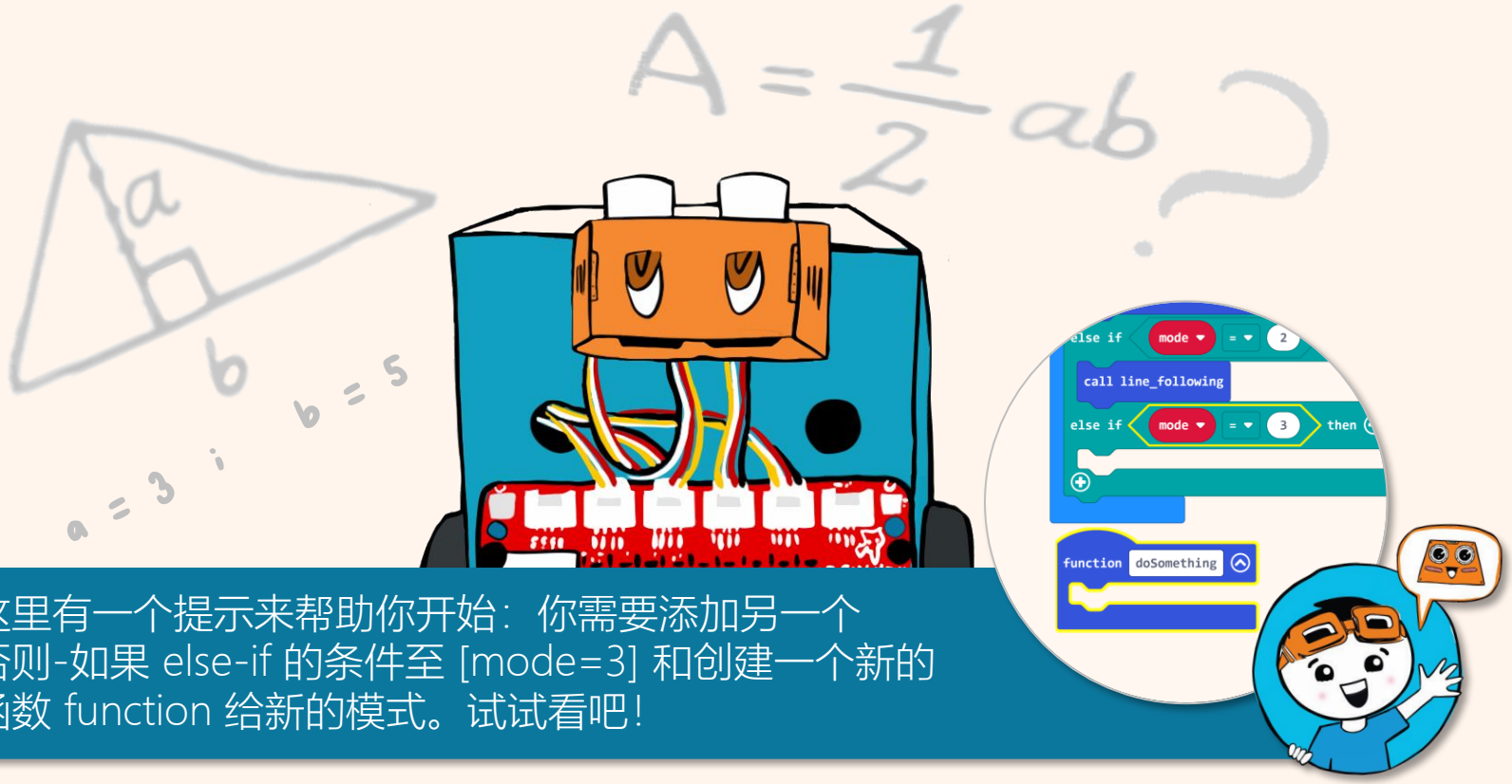
8 下载程序至你的 ZOOM:BIT，然后你就可以携带你的智能小车到所有的地方，并展现给你的朋友看了~

开始 (On Start)	播放声音 (hello)，显示笑脸和面向前方并打开两个车前灯。设置你的 mode 为 0 。
永远 (Forever)	永远检查 “mode” 若 Mode = 1，开启 Obstacle Avoidance function; 若 Mode = 2，开启 Line Following function 。
当徽标 (logo) 被按下 (或 ‘on logo down’ 若使用 micro:bit V1)	**添加 1 至 mode** ，停止移动并显示当前模式和相对应的图标：车形 (Mode 0)，心形 (Mode 1) 和，四方形 (Mode 2)。 若 mode 的数值不是 1 或 2 , 把 mode 设置回 0。
当 按钮 A 被按下时	向**右**转
当 按钮 B 被按下时	向**左**转
当 按钮 A 和 B 同时被按下时	向**前行**驶



为你准备了一个有趣的挑战！

你可以教导 ZOOM:BIT 一个新的技巧吗？ 一起尝试编程 ZOOM:BIT 来解答数学题目与公式吧！试着添加一个额外的 “mode” 到前面的程序里。



这里有一个提示来帮助你开始：你需要添加另一个否则-如果 else-if 的条件至 [mode=3] 和创建一个新的函数 function 给新的模式。试试看吧！

BONUS CHAPTER 附加章节

https://link.cytron.io
/zoombit-bonus-chapter

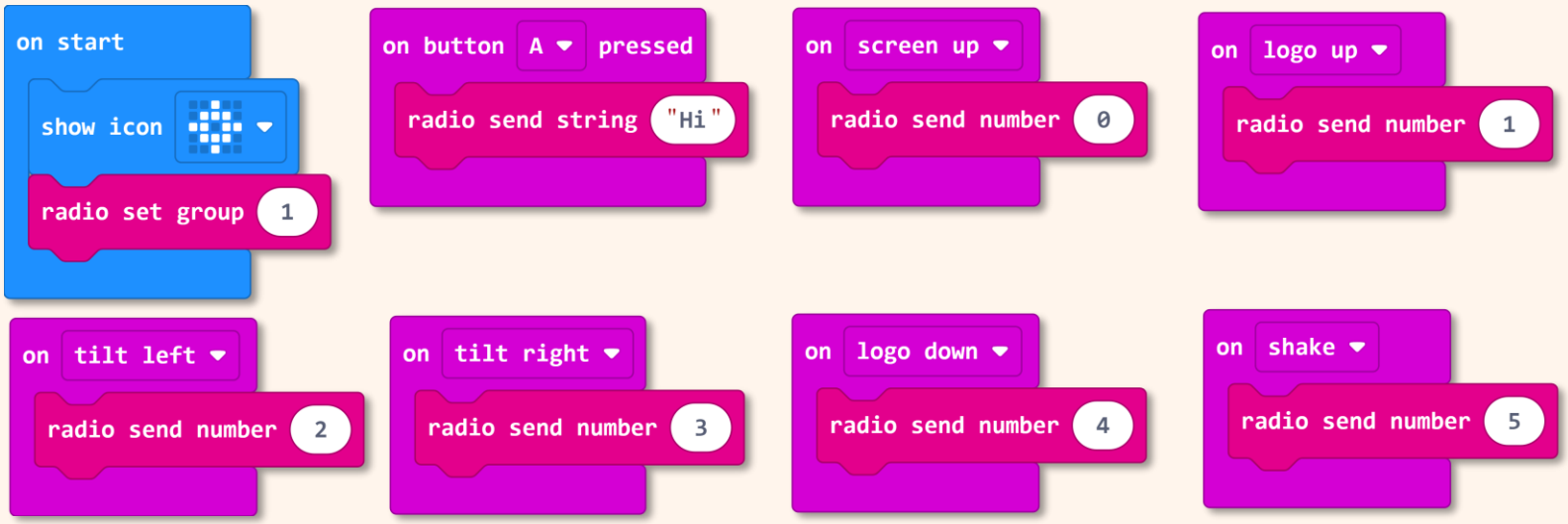
Roger , Roger . . . 你听到吗 ?

你是否知道？

ZOOM:BIT 上的 micro:bit 已配备 **无线电通讯 radio communication** 功能。换句话说，如果你还有另一台 micro:bit，你可以对其进行编程来让它变成一个遥控器以控制你的 ZOOM:BIT。让我们一起试试看吧！

1 续写以下程序并下载至将会成为**遥控器**的 micro:bit 板。
你可以到以下分类抽屉寻找你所需要的编码块。

Basic | Input | Radio

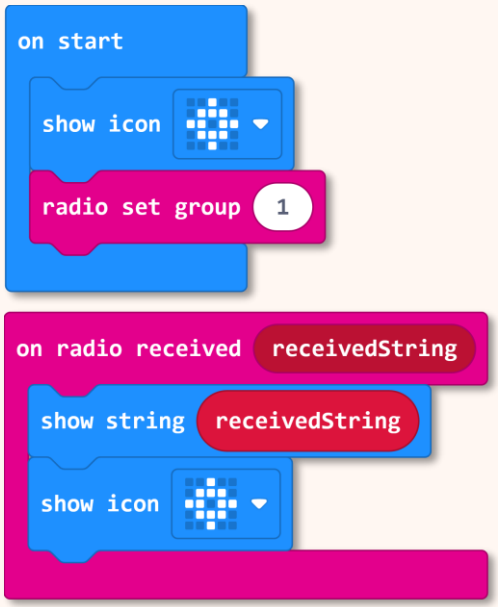


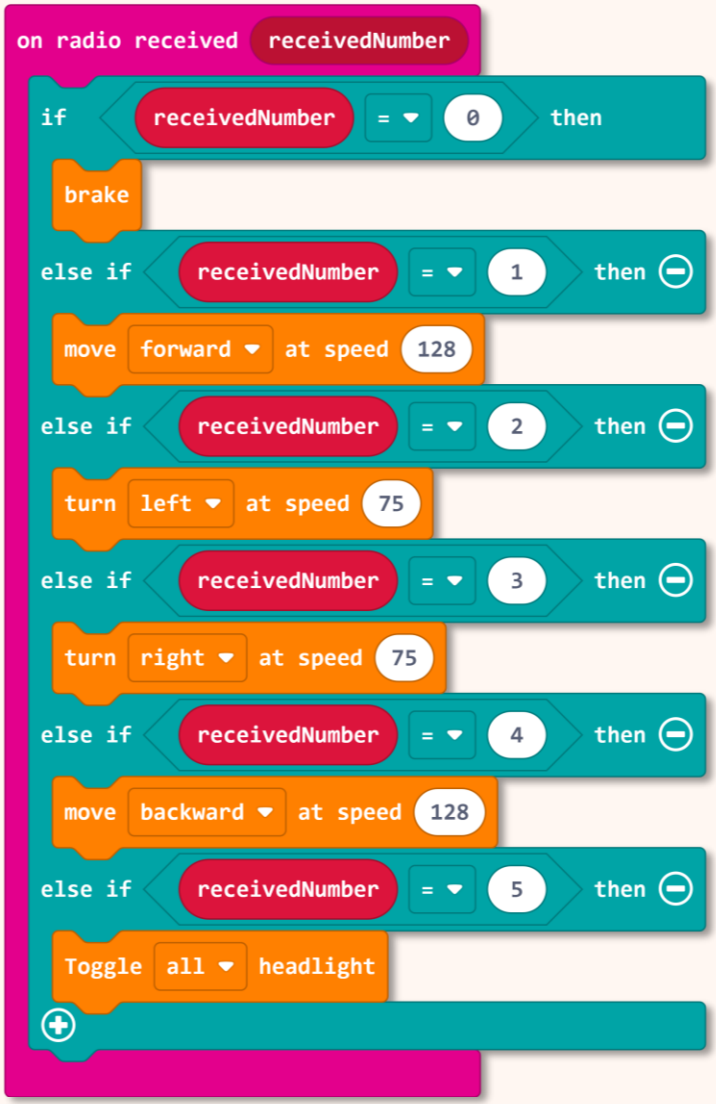
2 在你的 MakeCode Editor 中创建一个新项目。 然后添加 ZOOM:BIT 的扩展。*(你可以参考第 44-45 页)*

3 续写以下程序来使 ZOOM:BIT 能够接受来自遥控器的指令。

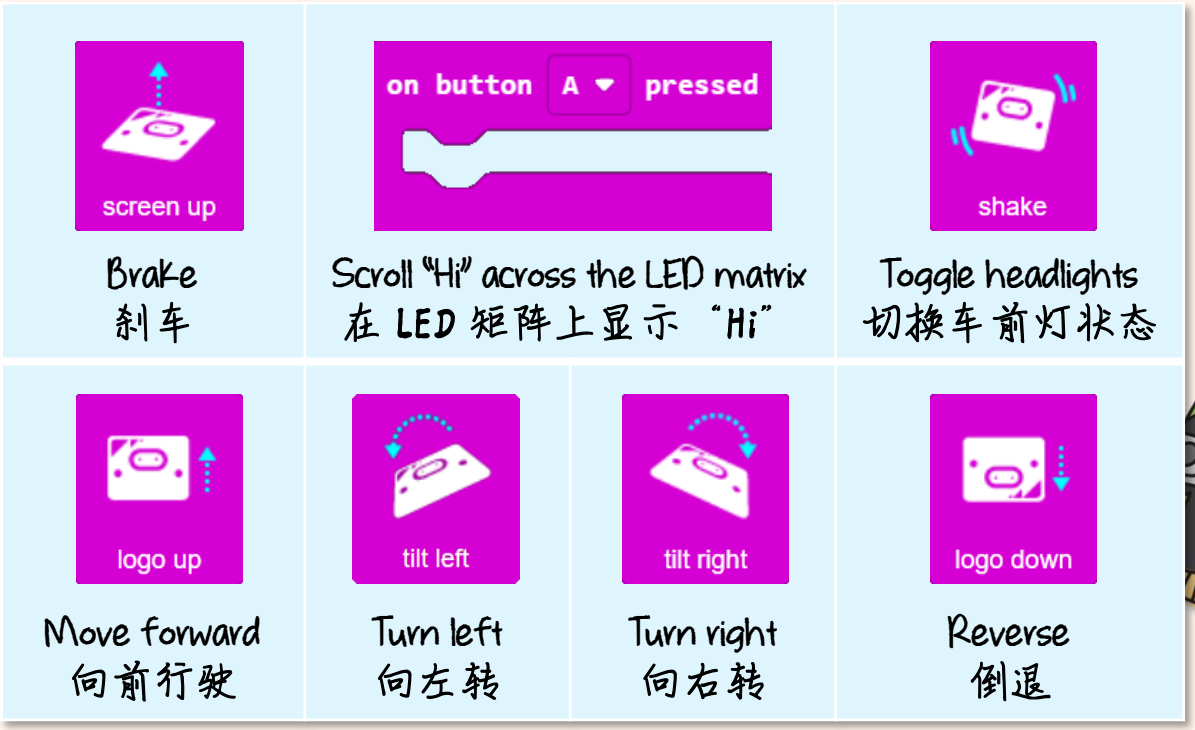
我们需要设置 micro:bit (遥控器) 以及 ZOOM:BIT 使用相同的无线设置组 radio group，以便它们可以互相发送和接受来自对方的信号。

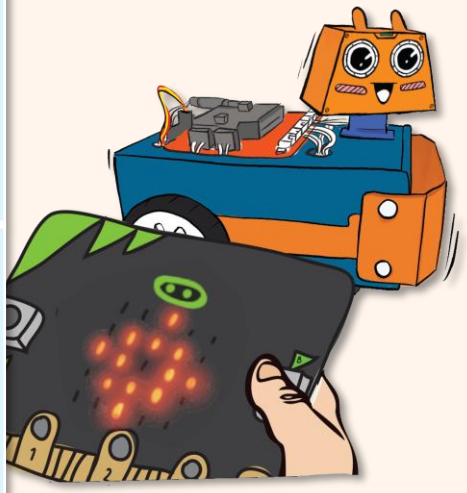
举个例子，我们将它们设置到第 1 组。





4 下载以下程序至 ZOOM:BIT 。打开 micro:bit（遥控器）和 ZOOM:BIT 的电源键。





现在你可以操控你的 ZOOM:BIT 来巡逻你的房间了。祝你玩的开心！

为你准备了一个有趣的挑战！

修改刚才的程序并添加更多新指令给予 ZOOM:BIT 来执行。
可能是传达一个秘密讯息或者运送一份礼物。
尝试给你的家庭成员和朋友一个惊喜 - 那就是把自己藏起来，
然后操控你的 ZOOM:BIT 去跟他们进行互动。

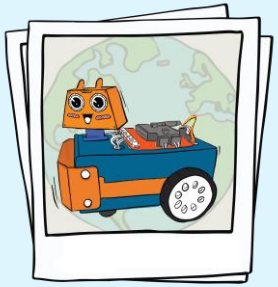
这里有一个提示来帮你开始。你可以添加 [on button B pressed]【当按钮 B 被按下时】和 [on button A+B pressed] 【当按钮 A+B 被按下时】到遥控器 (micro:bit) 的程序。
你也需要添加新的 否则-如果 else-if 条件到 ZOOM:BIT 的程序。

我与 ZOOM:BIT 的学习日记

我于_____年____月____日，成功完成搭建我的 ZOOM:BIT。
我也成功探索和研究册子里的课程及完成所有挑战。

第一章

显示文字与动画



课程与挑战
完成日期：

验证者：

第二章

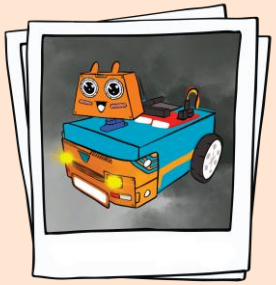
演奏简单的旋律

课程与挑战
完成日期：

验证者：

第三章

黑暗时，
打开车前灯

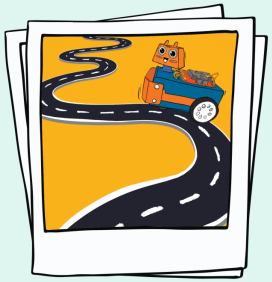


课程与挑战
完成日期：

验证者：

第四章

按命令
行驶与转弯

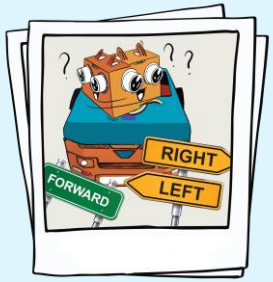


课程与挑战
完成日期：

验证者：

第五章

使用 RGB LED 灯
发出信号



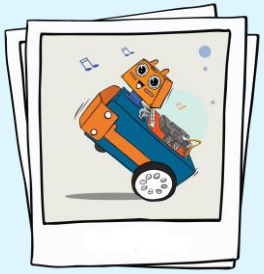
课程与挑战
完成日期：

验证者：

我与 ZOOM:BIT 的学习日记

第六章

控制头部
动作与角度

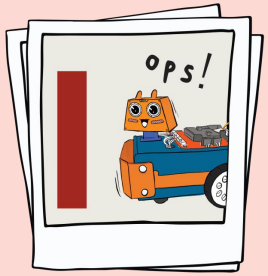


课程与挑战
完成日期：

验证者：

第七章

探测与避开
障碍物

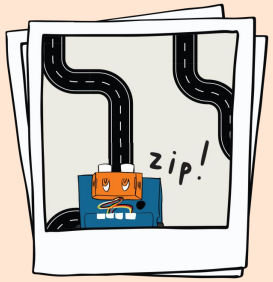


课程与挑战
完成日期：

验证者：

第八章

跟着路线行驶
保持在线上

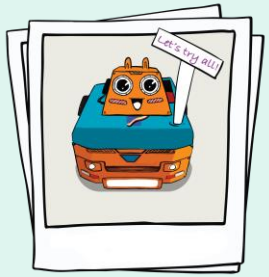


课程与挑战
完成日期：

验证者：

第九章

从一种模式
切换到另一种模式



课程与挑战
完成日期：

验证者：

附加章节

遥控

课程与挑战
完成日期：

验证者：

P/S 你可以请一位老师或家长与你共同检查及验证你的学习之路。

来自 Cytron rero EDUteam 的小帖士

Woohoo ... 恭喜你！你成功搭建自己的 ZOOM:BIT 智能小车，也与 ZOOM:BIT 一起学习到了编码以及完成所有的挑战。你做得真棒！我们希望你在此过程中能获得不少的乐趣。

那么下一步是什么呢？你可以浏览 www.cytron.io 来探索和获取附加传感器或零件来客制自己的智能小车。你想不想使用 Grove OLED Display 来显示更丰富的图像？又或者是加上更多伺服电机来构成一个机械手臂？继续快乐地探索无限的可能吧！

https://t.me/zoombit_support

你可以在 Telegram t.me./zoombit_support 和我们共享你的 ZOOM:BIT 项目及学习之旅。我们将非常乐意听到你的消息。Cheers~

亚当 和 安娜 上

ISBN 978-967-19475-1-7