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

SunFounder PiCrawler 机器人

SunFounder

2023 年 09 月 25 日

Contents

感谢您选择我们的 PiCrawler。

CHAPTER 1

元件清单和组装说明

您需要先根据清单检查是否有缺少或损坏的组件。如果有任何问题，请联系我们，我们会尽快解决。
请按照 PDF 上的步骤进行组装。
如果舵机已经上电，请勿强行转动舵机，以免损坏。

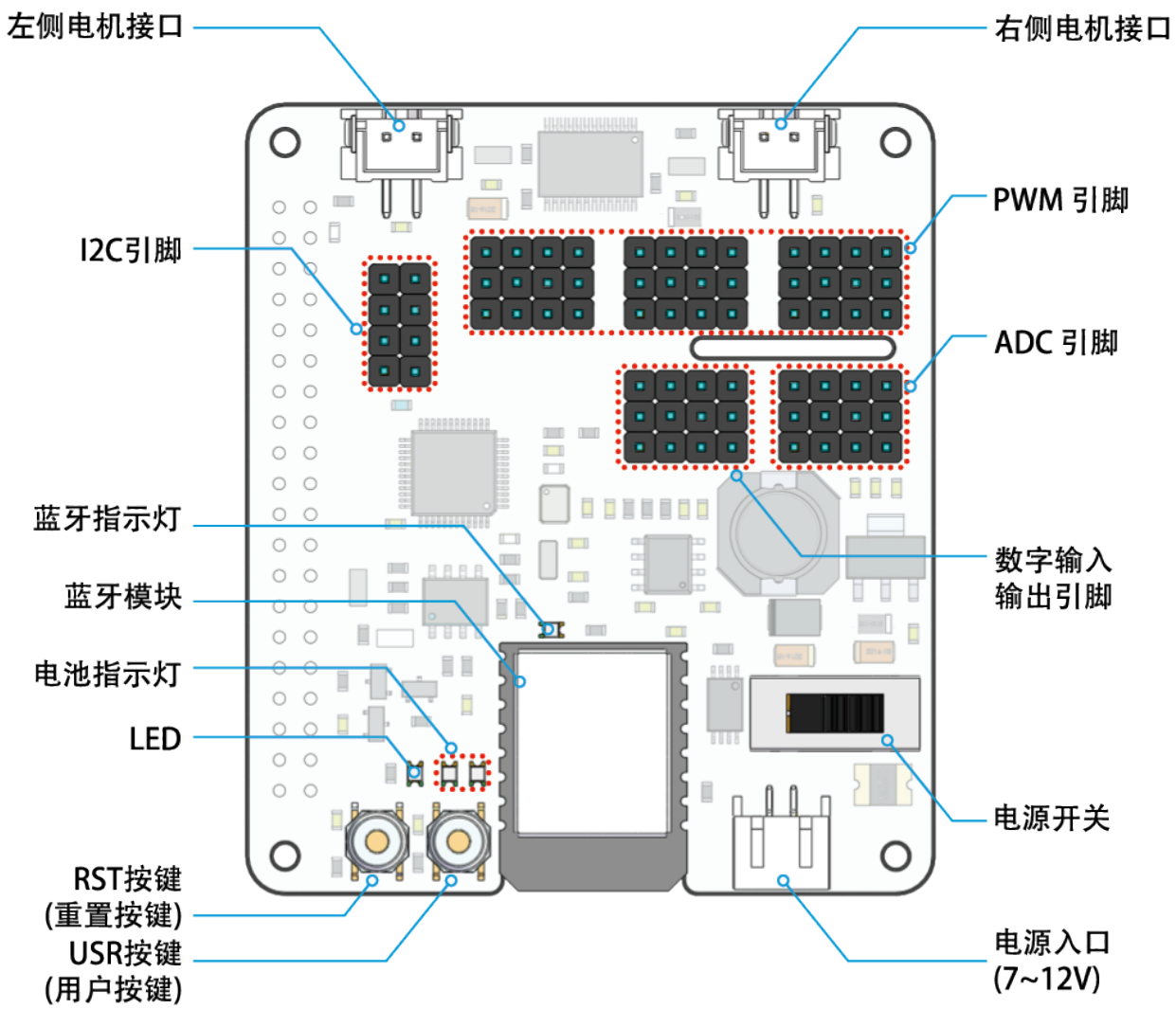
注解:

1. 组装前需要购买 2 节 18650 电池并充满电，参见关于电池.
2. Robot HAT 不能给电池充电，需要同时购买充电器。

- `PiCrawler 组装教程.pdf`

CHAPTER 2

关于 Robot HAT



RST 按钮

- 短按 RST 按钮将导致所有正在运行的程序重置。
- 长按 RST 按钮直到 LED 亮起，然后松开将断开 Robot HAT 的蓝牙芯片。

USR 按钮

- USR 按钮的功能可通过编程进行配置。（按下导致输入 `0` ，松开产生 `1` 输入）

LED

- 通过编程配置（输出 `1` 打开 LED，输出 `0` 关闭 LED。）

电池指示灯

- 电池电压高于 7.8V 将点亮两个 LED 指示灯。电池电压在 6.7V 到 7.8V 之间只会点亮一个 LED，低于 6.7V 的电压将关闭两个 LED。

蓝牙指示灯

- 蓝牙指示灯 LED 将在蓝牙连接稳定时保持亮起，并在信号传输期间快速闪烁。如果蓝牙断开连接，LED 将每隔 1 秒闪烁一次。

CHAPTER 3

玩转 Python

如果你想用 python 编程，那么你需要学习一些基本的 Python 编程技巧和树莓派的基础知识，请先根据 Python 快速指南配置树莓派。

3.1 Python 快速指南

本节教大家如何安装树莓派操作系统，配置树莓派 wifi，远程访问树莓派运行相应代码。

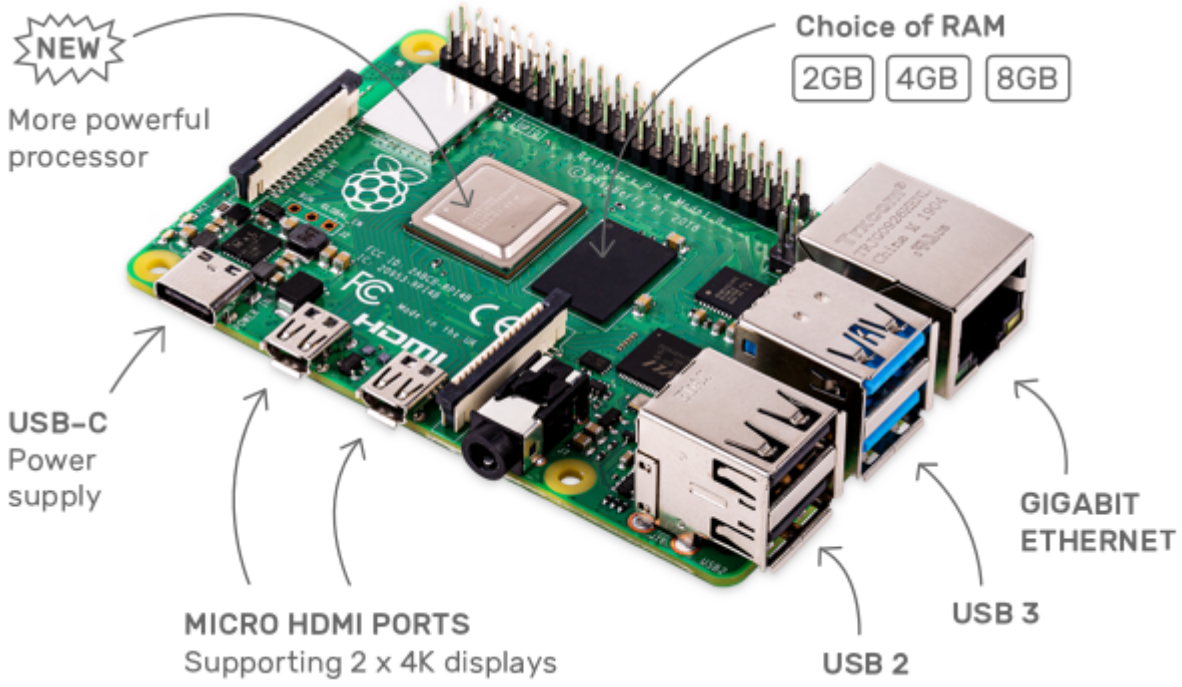
如果您熟悉树莓派并且可以成功打开命令行，那么您可以跳过前 3 部分，然后完成最后一部分。

3.1.1 我们需要什么？

所需组件

树莓派

树莓派是一款低成本、信用卡大小的计算机，可插入进入电脑显示器或电视，并使用标准键盘和鼠标。这是一款功能强大的小型设备，可让所有年龄段的人探索计算，并学习如何使用 Scratch 和 Python 等语言进行编程。



电源适配器

为了连接到电源插座，树莓派有一个微型 USB 端口（在许多手机上都可以找到）。您将需要一个提供至少 2.5 安培电流的电源。

微型 SD 卡

您的树莓派需要一张 Micro SD 卡来存储其所有文件和树莓派操作系统。您需要一张容量至少为 8 GB 的微型 SD 卡。

可选组件

屏幕

要查看树莓派的桌面环境，需要使用的屏幕可以是电视屏幕，也可以是电脑显示器。如果屏幕有内置扬声器，Pi 通过它们播放声音。

鼠标和键盘

使用屏幕时，还需要一个 USB 键盘和一个 USB 鼠标。

HDMI

树莓派有一个 HDMI 输出端口，与大多数现代电视和计算机显示器的 HDMI 端口兼容。如果您的屏幕只有 DVI 或 VGA 端口，则需要使用合适的转换线。

外壳

你可以把树莓派放在一个盒子里；通过这种方式，您可以保护您的设备。

声音或耳机

树莓派配备了一个 3.5mm 左右的音频接口，可以在你的屏幕没有内置扬声器或者没有屏幕操作的情况下使用。

3.1.2 安装操作系统

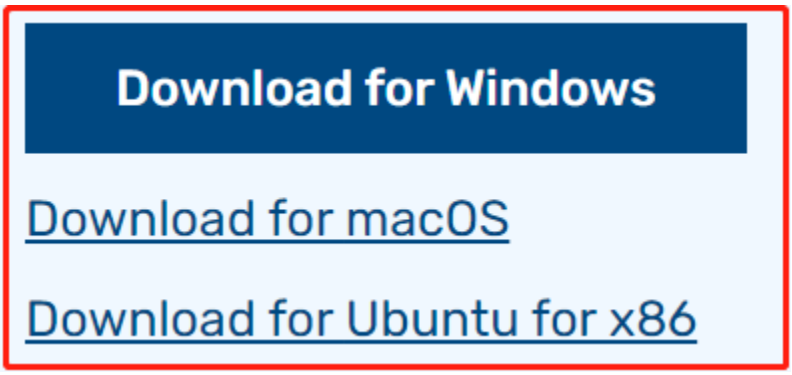
必需组件

任意树莓派	1 * 个人计算机
1 * 微型 SD 卡	

第 1 步

树莓派开发了一个图形 SD 卡写入工具，适用于 Mac OS、Ubuntu 18.04 和 Windows，对于大多数用户来说是最简单的选择，因为它会下载映像并将其自动安装到 SD 卡。

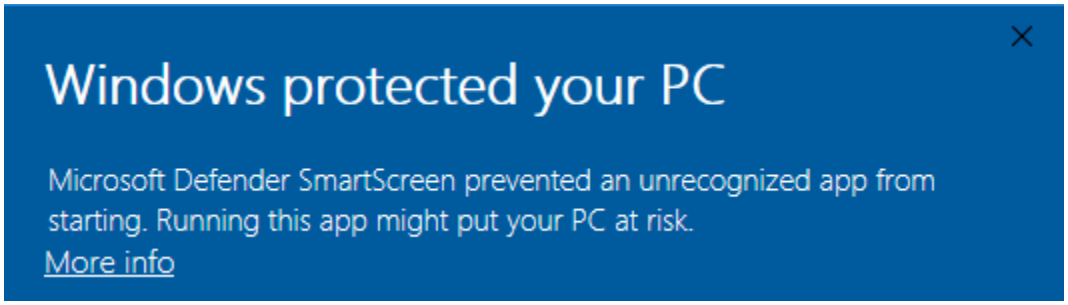
访问下载页面：https://www.raspberrypi.org/software/。单击与您的操作系统匹配的 Raspberry Pi Imager 链接，下载完成后，单击它以启动安装程序。



第 2 步

当您启动安装程序时，您的操作系统可能会尝试阻止您运行它。例如，在 Windows 上，我收到以下消息：

如果出现此消息，请点击 **更多信息**，然后点击 **仍然运行**，然后按照说明安装 Raspberry Pi Imager。



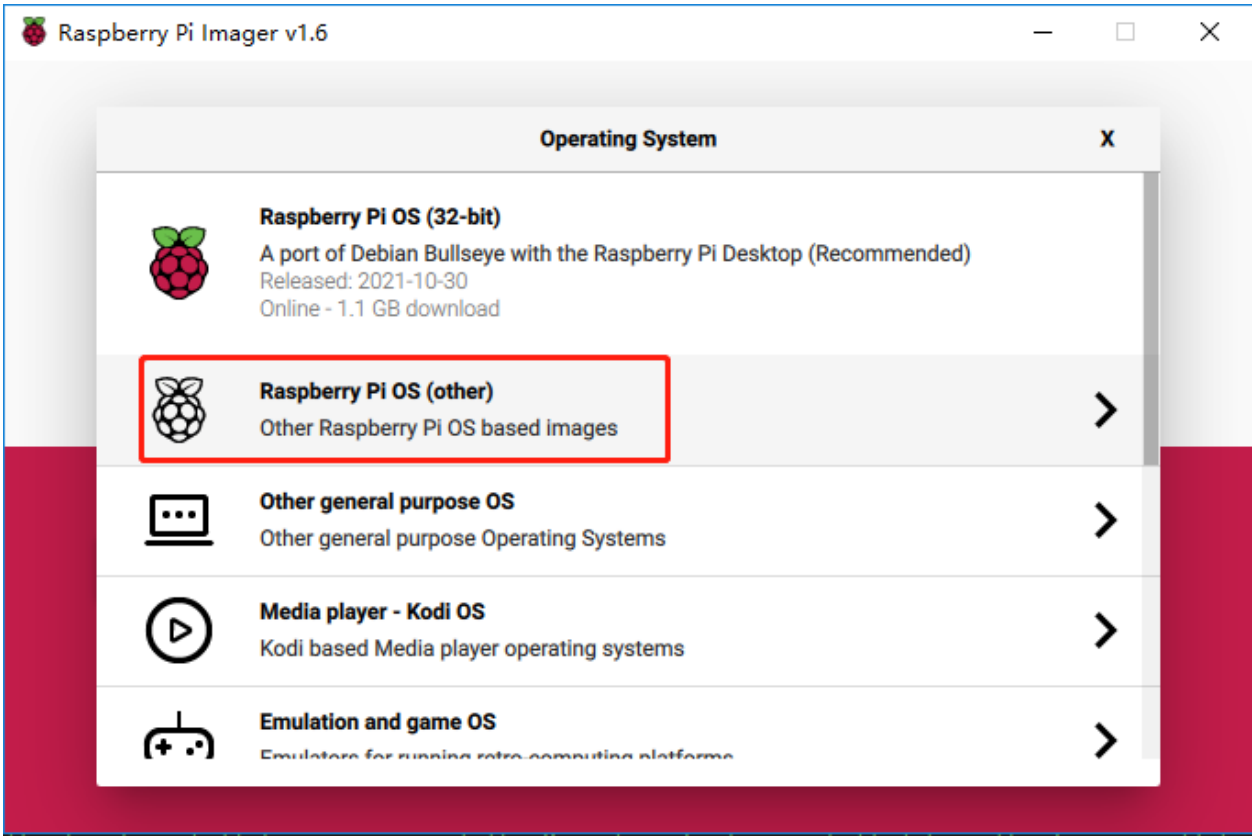
第 3 步

将 SD 卡插入计算机或笔记本电脑的 SD 卡插槽。

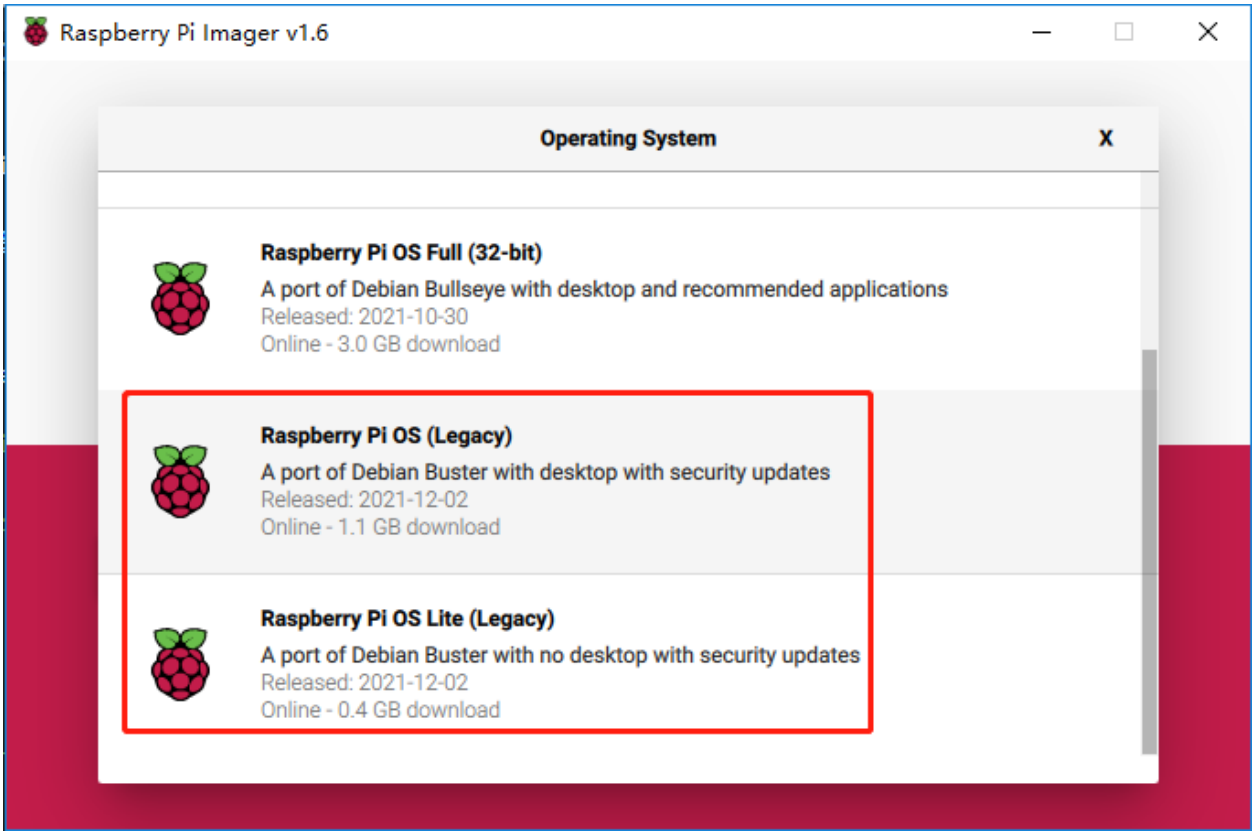
第 4 步

> **警告：** Raspberry Pi OS 升级到 **Debian Bullseye** 后，会导致有些功能不能使用，建议还是继续使用 **Debian Buster** 版本。

在 Raspberry Pi Imager 中，点击 **CHOOSE OS -》 Raspberry Pi OS(other)**。

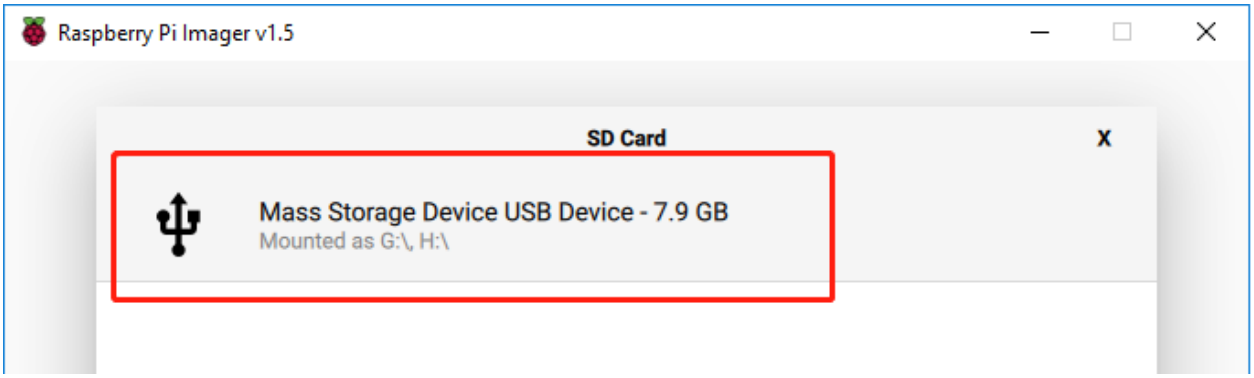


将新打开的页下拉到最后面，你会看到 **Raspberry Pi OS(Legacy)** 和 **Raspberry Pi OS Lite(Legacy)**，这 2 个是对 Debian Buster 安全更新，它们之间的区别是带不带桌面。建议安装 **Raspberry Pi OS(Legacy)**，这个带桌面的系统。



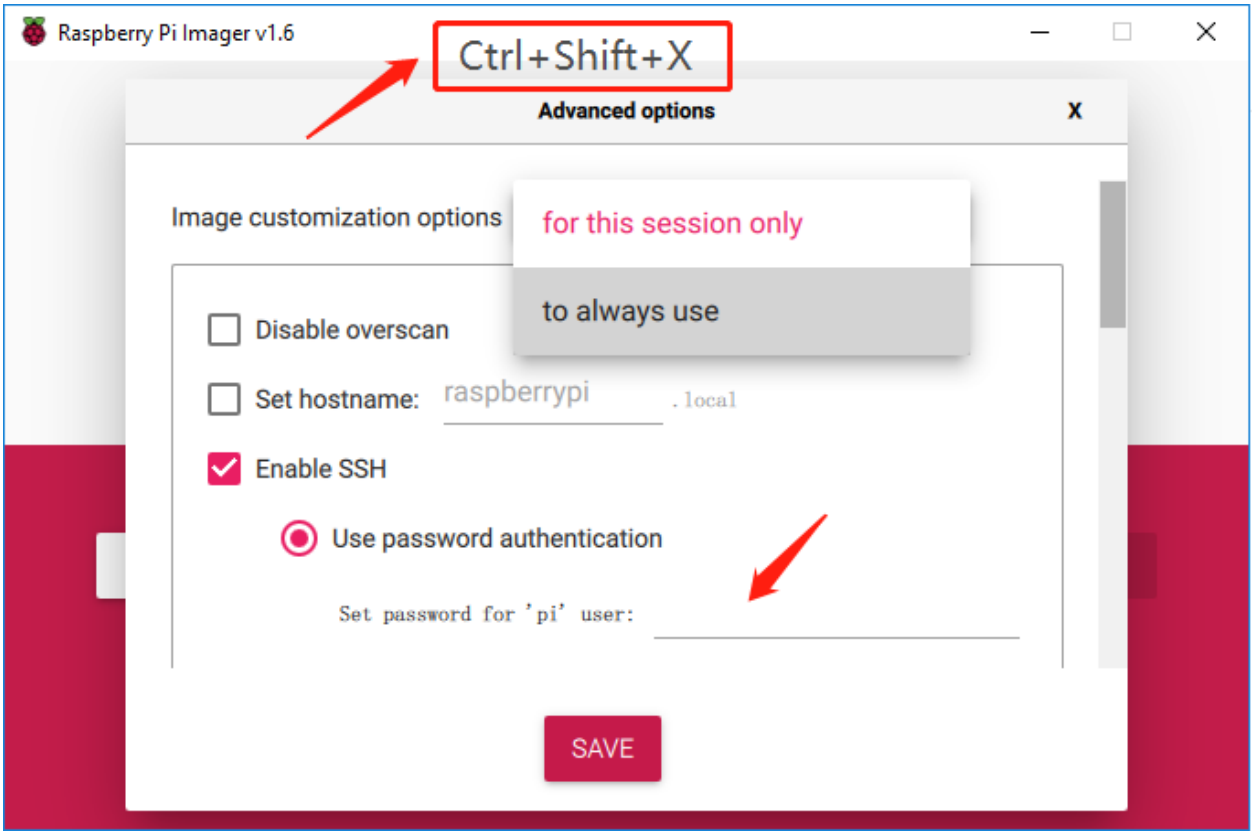
第 5 步

选择您正在使用的 SD 卡。



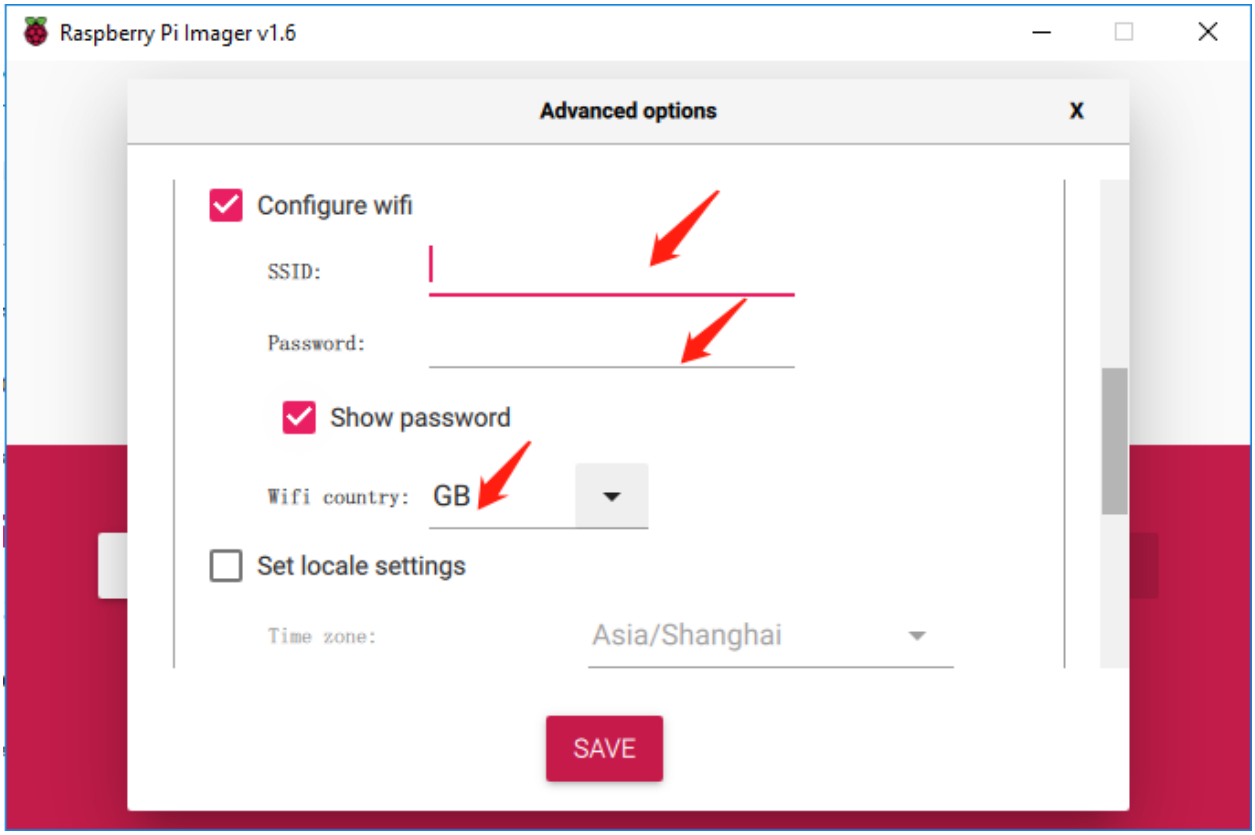
第 6 步

按 **Ctrl+Shift+X** 或者点击 **设置**按钮来打开 **高级选项**页面启用 SSH 和配置 wifi，这 2 项必须设置，其他取决于你的选择。您可以选择始终使用此图像自定义选项。



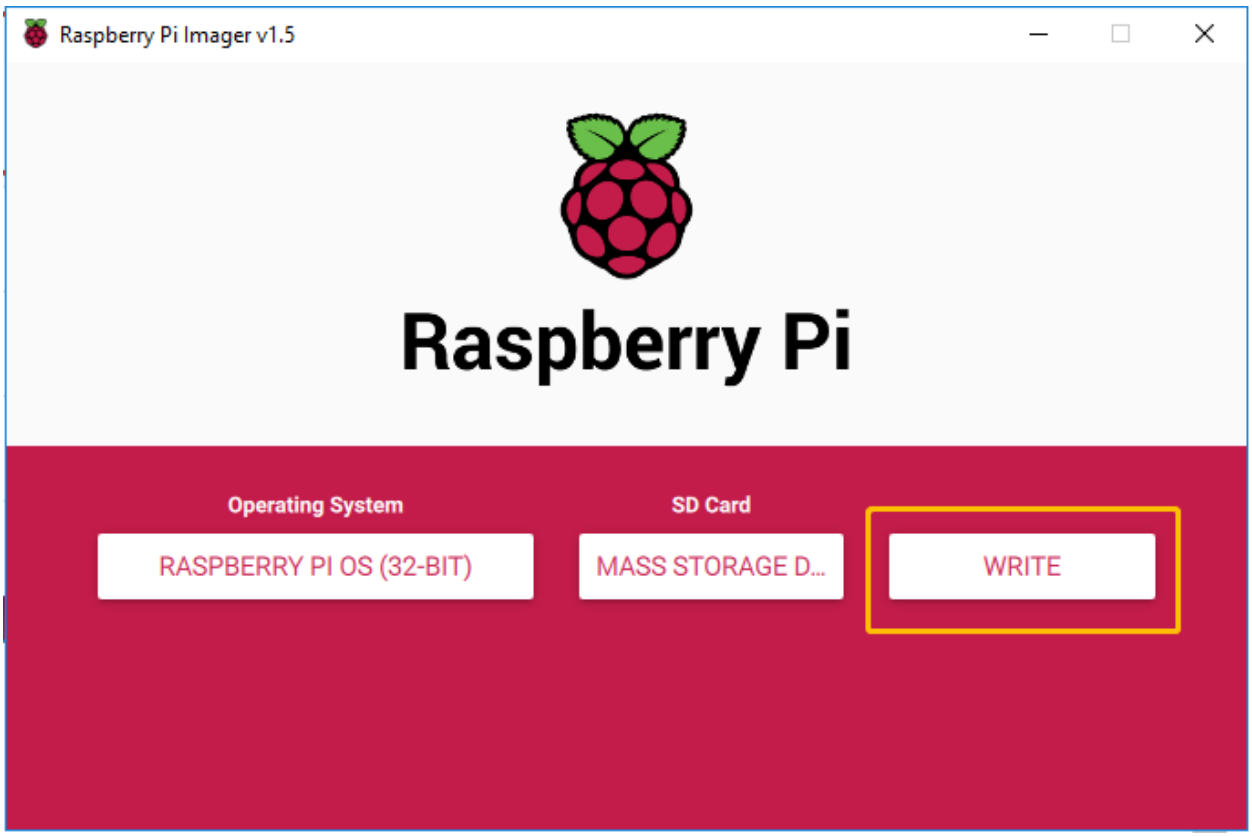
然后向下滚动以完成 wifi 配置并单击 **SAVE** 。

注解： **wifi country** 选择 **CN**。



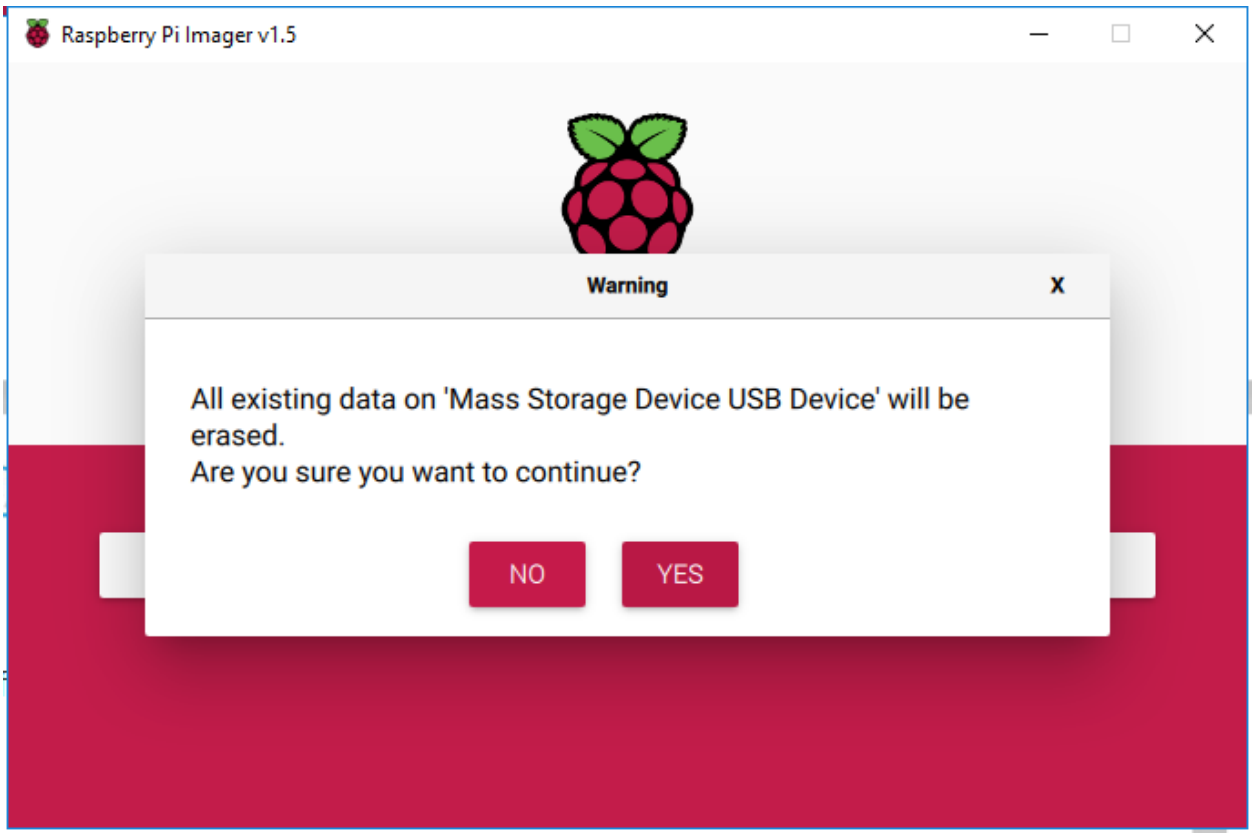
第 7 步

单击 **WRITE** 按钮。



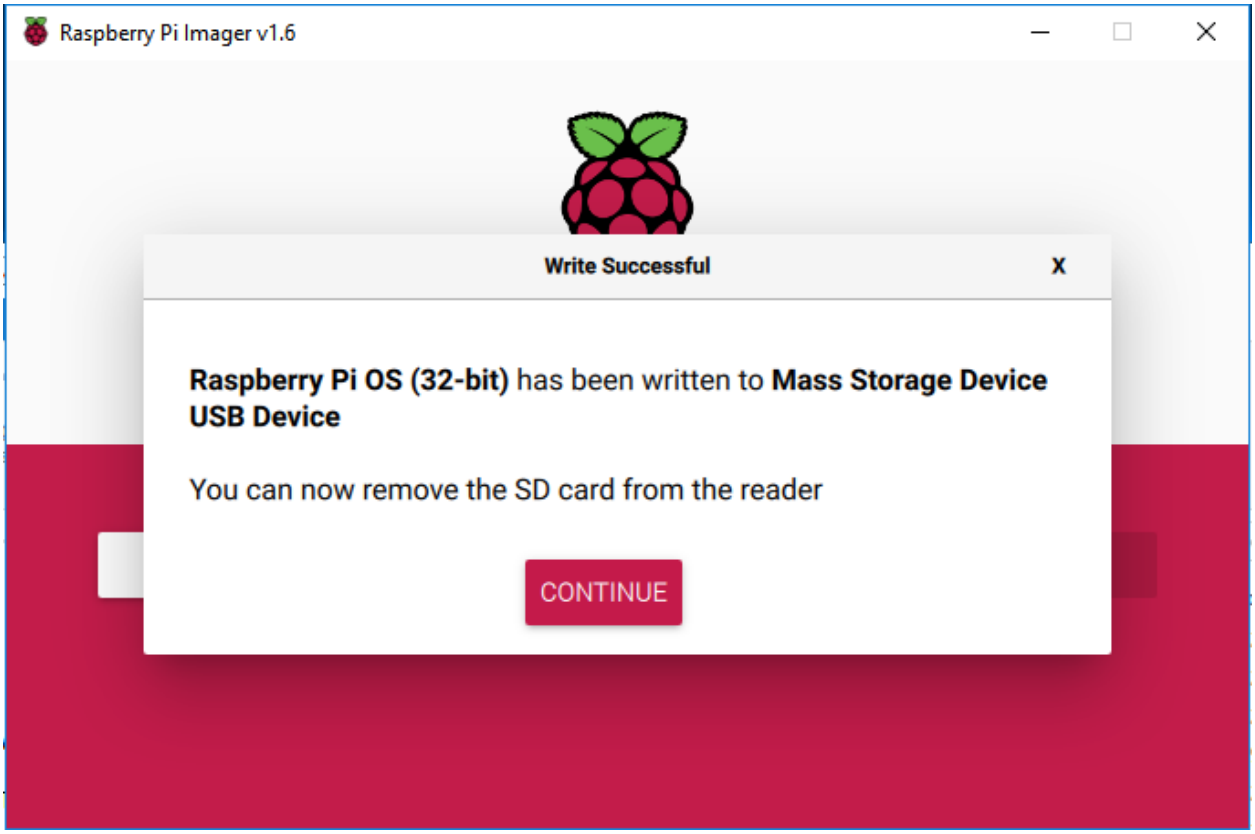
第 8 步

如果您的 SD 卡上当前有任何文件，您可能希望先备份这些文件以防止永久丢失它们。如果没有要备份的文件，请单击 **YES**。



第 9 步

等待一段时间后，会出现如下窗口，代表写入完成。



3.1.3 设置你的树莓派

如果你有屏幕

如果你有一个屏幕，你会很容易在屏幕上操作树莓派。

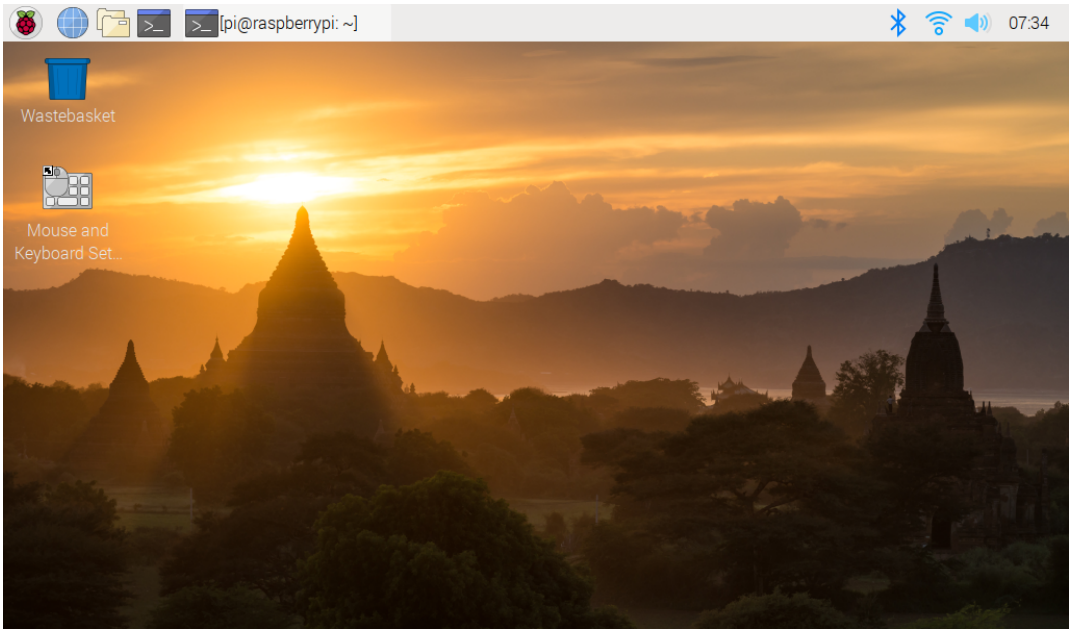
必需组件

任意树莓派	1 * 电源适配器
1 * Micro SD 卡	1 * 屏幕电源适配器
1 * HDMI 线	1 * 屏幕
1 * 鼠标	1 * 键盘

1. 将您在 Raspberry Pi OS 上设置的 SD 卡插入到树莓派底部的 micro SD 卡插槽中。
2. 插入鼠标和键盘。
3. 将屏幕连接到树莓派的 HDMI 端口，并确保您的屏幕已插入壁式插座并已打开。

> **注解：** 如果您使用的是树莓派 4，则需要将屏幕连接到 HDMI0（最靠近电源输入端口）。

4. 使用电源适配器为树莓派供电。几秒钟后，将显示 Raspberry Pi OS 桌面。



如果你没有屏幕

如果没有显示器，可以远程登录树莓派，但在此之前，您需要获取树莓派的 IP。

获取 IP 地址

树莓派连接 WIFI 后，我们需要获取它的 IP 地址。知道 IP 地址的方法有很多种，下面列出了其中的两种。

1. 通过路由器检查

如果您有权限登录路由器（如家庭网络），您可以在路由器的管理界面查看分配给树莓派的地址。

树莓派操作系统的默认主机名是 **raspberrypi**，你需要找到它。（如果你使用的是 ArchLinuxARM 系统，请找 alarmpi。）

2. 网段扫描

您还可以使用网络扫描来查找树莓派的 IP 地址。您可以应用软件，**Advanced IP scanner** 等。

扫描设置的 IP 范围，将显示所有已连接设备的名称。同样，树莓派操作系统的默认主机名是 **raspberrypi**，如果你没有修改过的话。

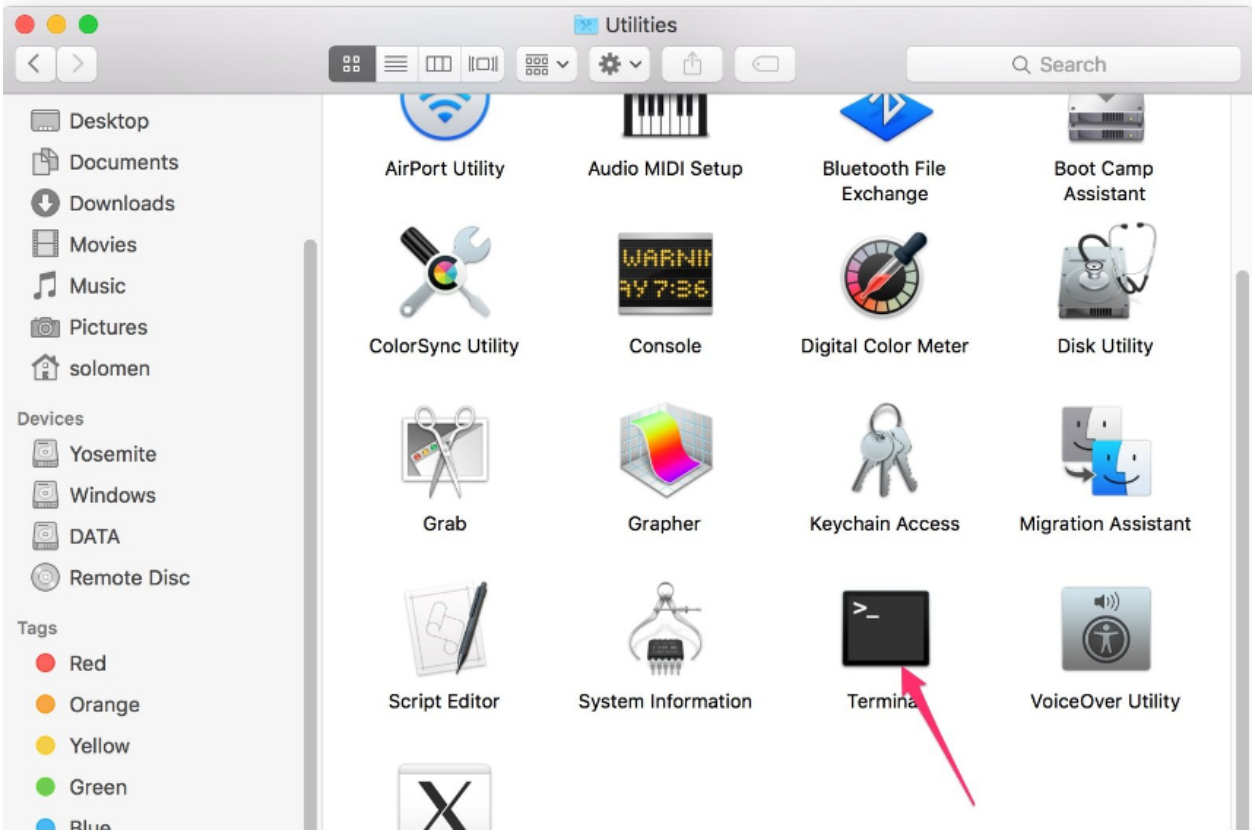
使用 SSH 远程控制

我们可以通过应用 SSH 打开树莓派的 Bash Shell。Bash 是 Linux 的标准默认 shell。Shell 本身是一个用 C 编写的程序，它是连接客户和 Unix/Linux 的桥梁。此外，它可以帮助完成大部分所需的工作。

适用于 Linux 或/Mac OS X 用户

第 1 步

进入 **Applications -> Utilities**，找到 **Terminal**，然后打开它。



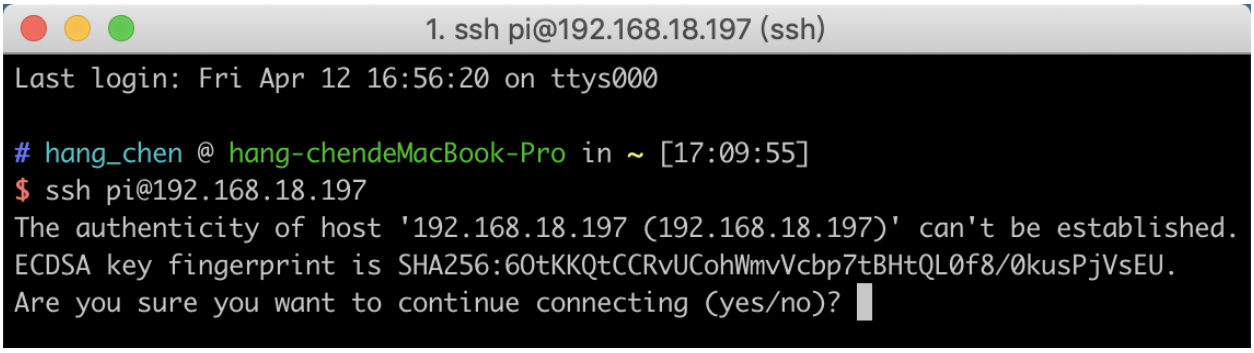
第 2 步

输入 **ssh pi@ip_address** 。”pi” 是您的用户名，”ip_address” 是您的 IP 地址。例如:

```
ssh pi@192.168.18.197
```

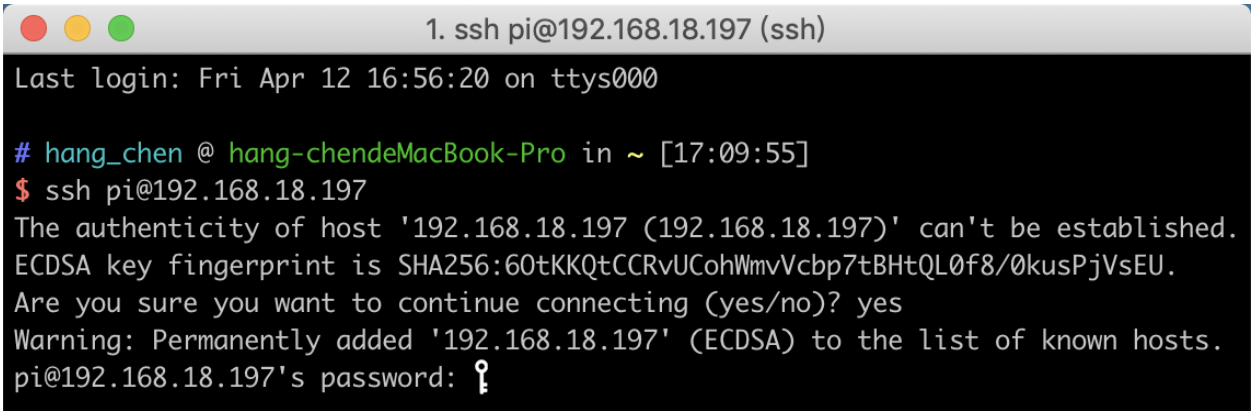
第 3 步

输入”yes”.



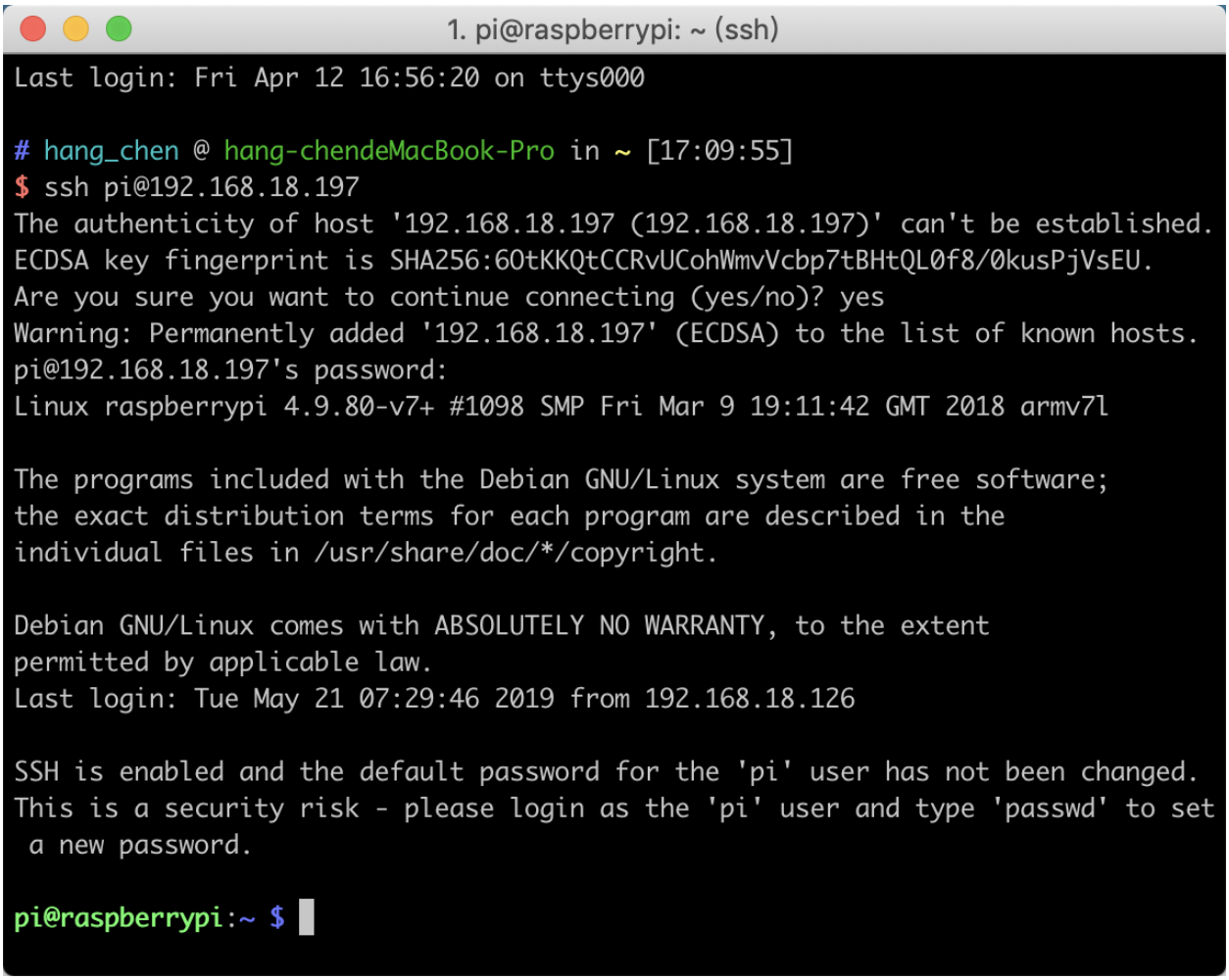
第 4 步

输入密码，默认密码为 **raspberry**。



第 5 步

我们现在已连接树莓派，并准备进行下一步。



注解： 当您输入密码时，字符不会相应显示在窗口中，这是正常的。您只需要输入正确的密码即可。

对于 Windows 用户

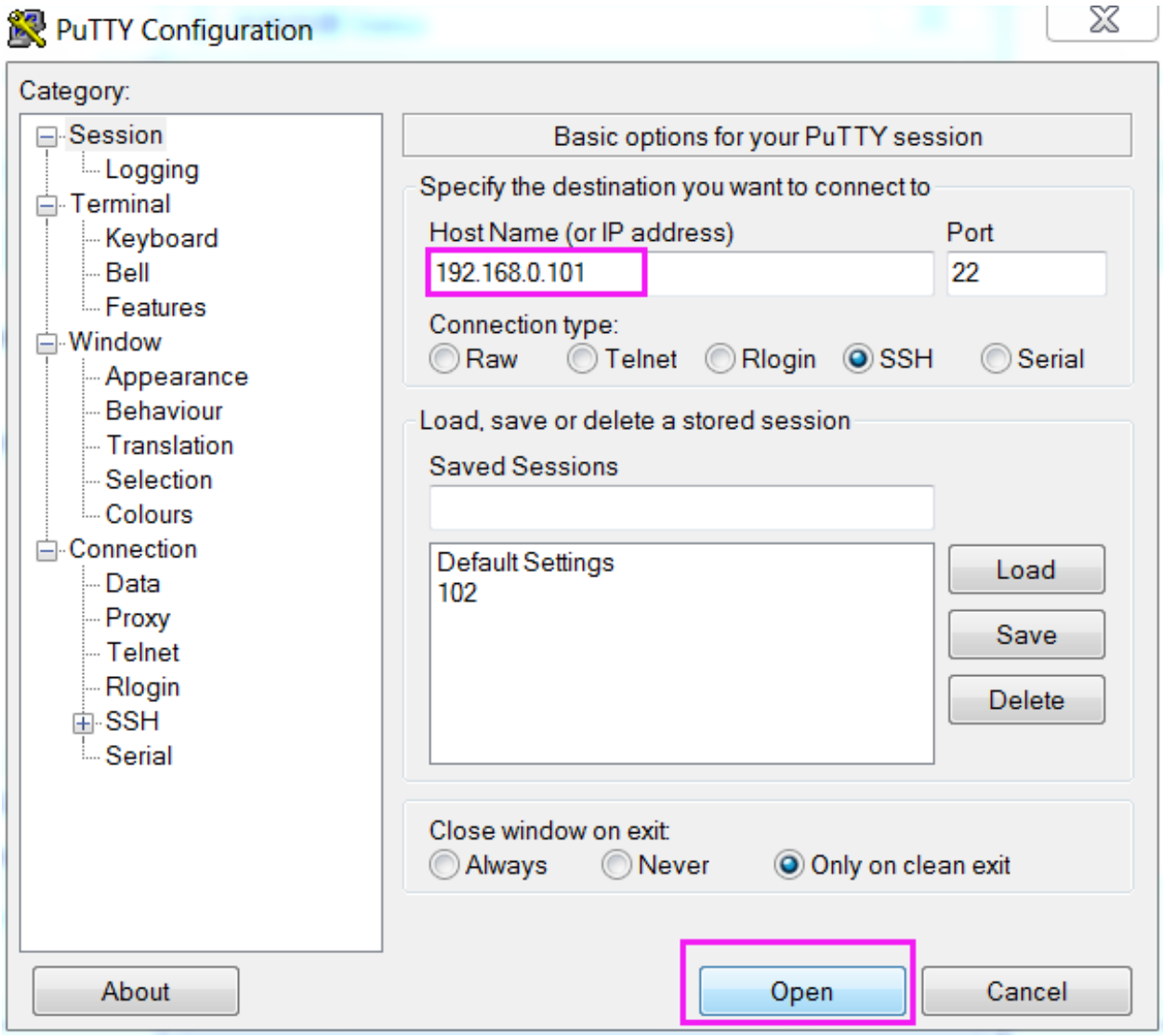
如果您是 Windows 用户，则可以通过某些软件的应用程序使用 SSH。在这里，我们推荐 **PuTTY**。

第 1 步

下载 **PuTTY**。

第 2 步

打开 PuTTY 并单击左侧树状结构上的 **Session**。在 **Host Name (or IP address)** 下的文本框中输入 RPi 的 IP 地址，**端口**下输入 **22** （默认为 22）。



第 3 步

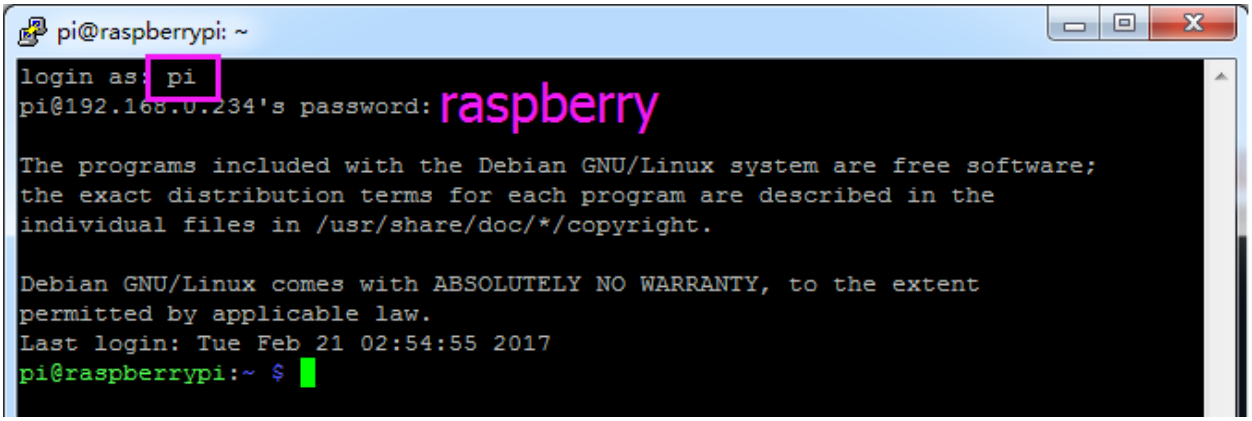
点击 **Open**。注意第一次用 IP 地址登录树莓派时，会提示安全提示。只需单击 **Yes**。

第四步

当 PuTTY 窗口提示 **login as:** 时，输入”pi”（树莓派的用户名）。提示 **password:** 时候，输入”raspberry”（默认密码）。

注解： 当您输入密码时，字符不会相应显示在窗口中，这是正常的。您只需要输入正确的密码即可。

如果 PuTTY 旁边出现 inactive，则表示连接已断开，需要重新连接。



第 5 步

在这里，我们连接了树莓派，是时候进行下一步了。

注解： 如果您对使用命令窗口控制树莓派不满意，还可以使用远程桌面功能，它可以帮助我们轻松管理树莓派中的文件。

有关如何执行此操作的详细信息，请参阅远程桌面。

3.1.4 下载并运行代码

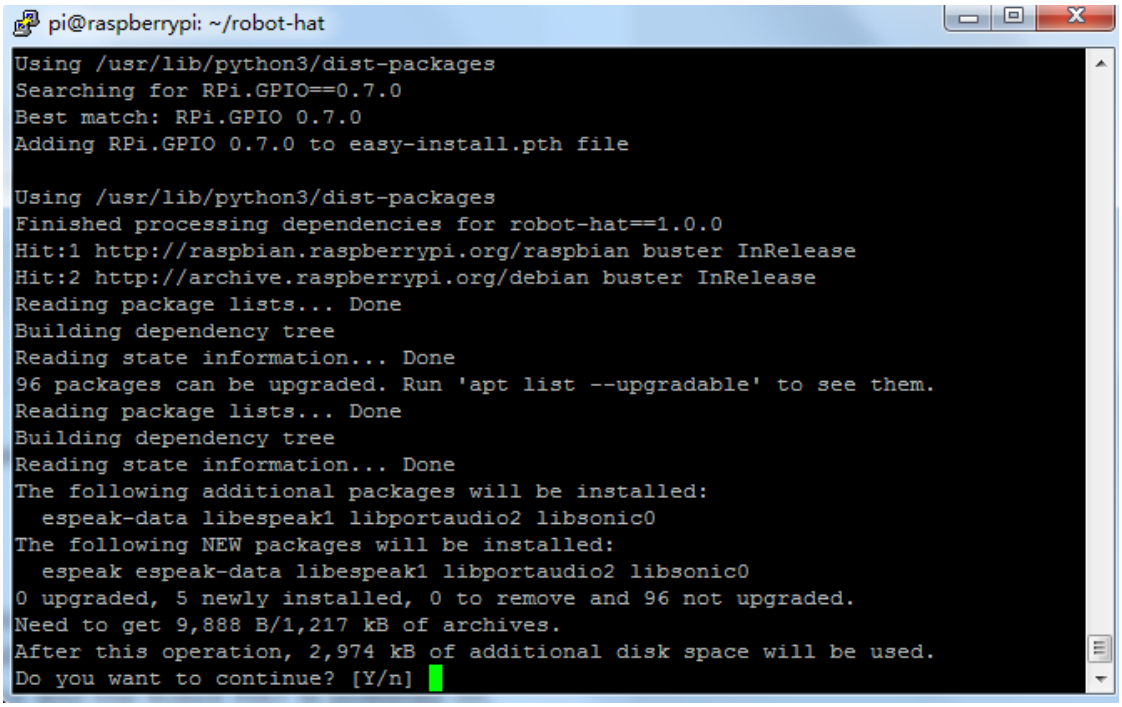
注解： 在下面的安装过程中，可能会由于网络问题导致失败，你需要参考*apt 和 pip 更换国内源* 来修改配置。

我们可以在命令行中通过 `git clone` 命令下载文件。

首先安装 `robot-hat` 库。

```
cd /home/pi/
git clone https://gitee.com/sunfounder/robot-hat.git
cd robot-hat
sudo python3 setup.py install
```

注解： 运行 `setup.py` 将下载一些必要的组件。由于网络问题，您可能无法下载成功。您可能需要重新下载。在这种情况下，输入 `Y` 并按 Enter.



然后下载代码并安装 `vilib` 库。

```
cd /home/pi/
git clone https://gitee.com/sunfounder/vilib.git
cd vilib
sudo python3 install.py
```

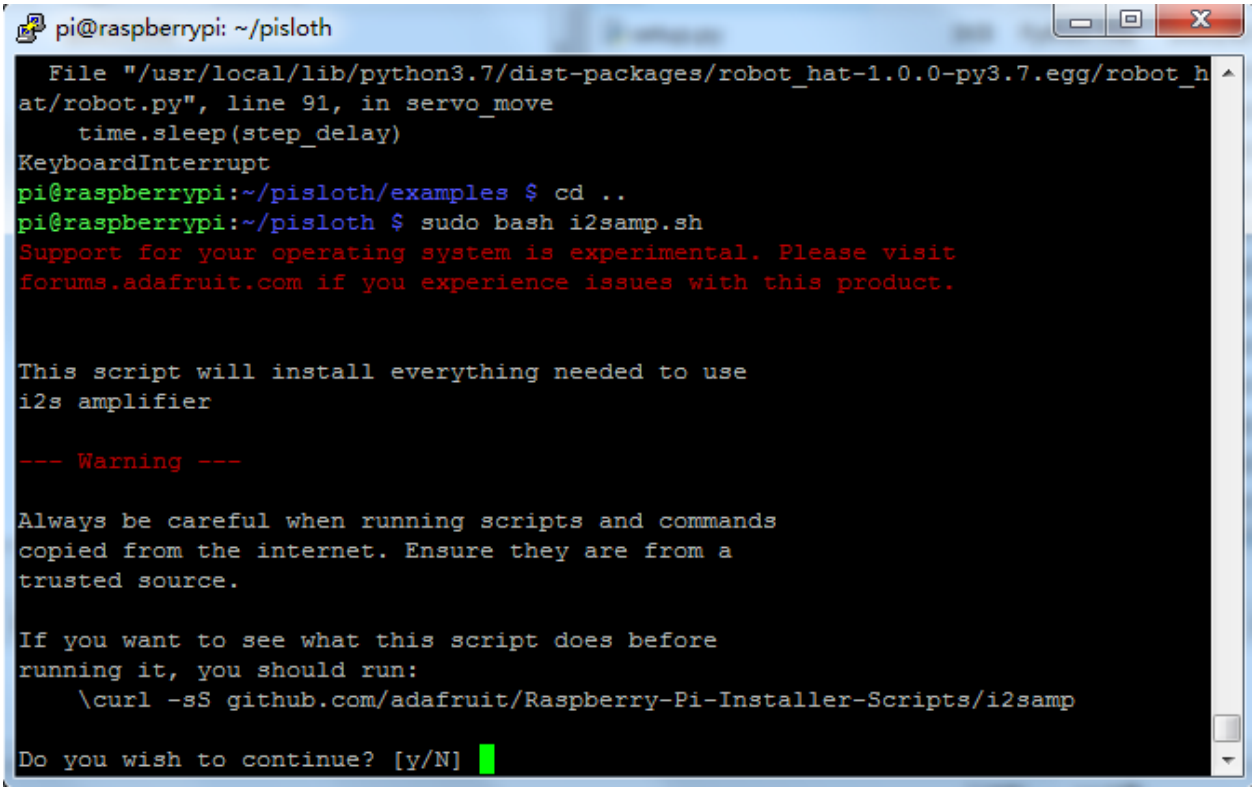
然后下载代码并安装 `picrawler` 库。

```
cd /home/pi/
git clone -b v2.0 https://gitee.com/sunfounder/picrawler.git
cd picrawler
sudo python3 setup.py install
```

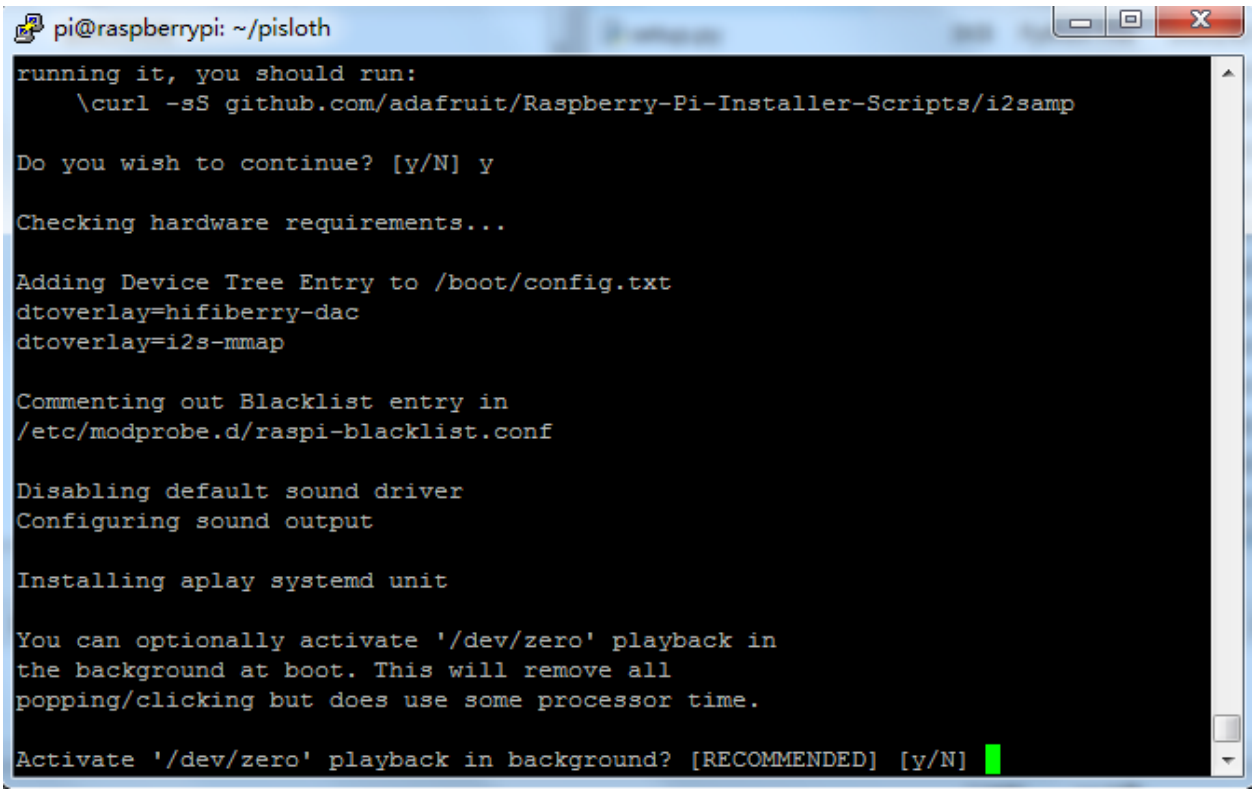
这一步需要一点时间，所以请耐心等待。

最后需要运行脚本 `i2samp.sh` 安装 i2s 功放所需的组件，否则它可能会没有声音。

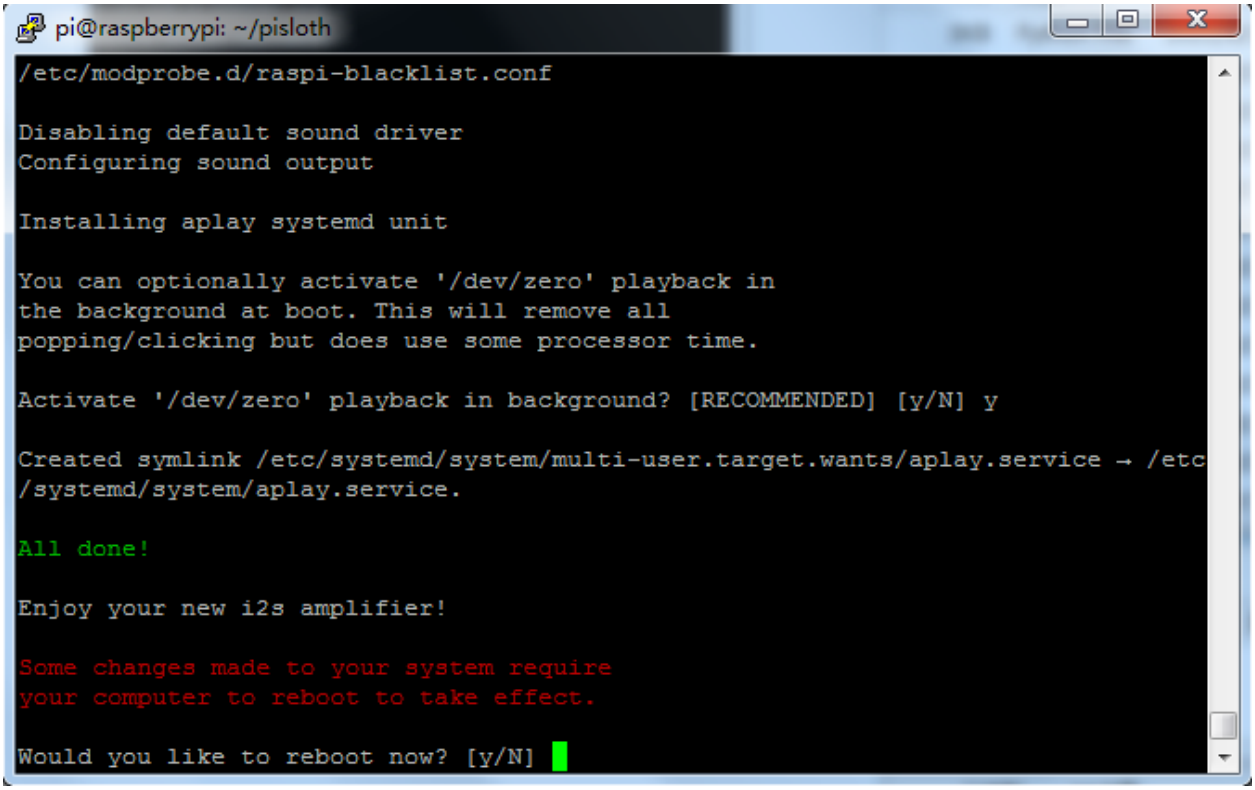
```
cd /home/pi/picrawler
sudo bash i2samp.sh
```



输入 y 并按 Enter 继续运行脚本。



输入 y 并按 Enter 让 `/dev/zero` 在后台运行。

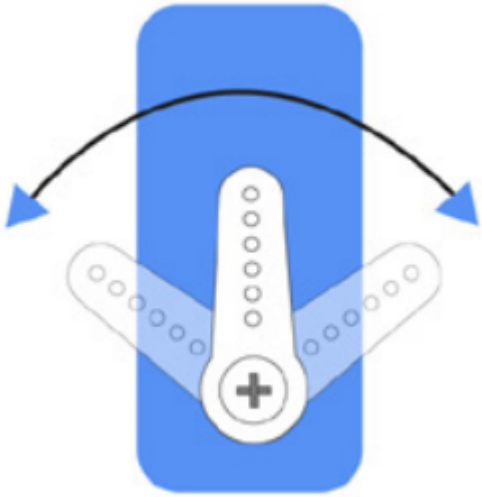


输入 y 并按 Enter 重新启动机器。

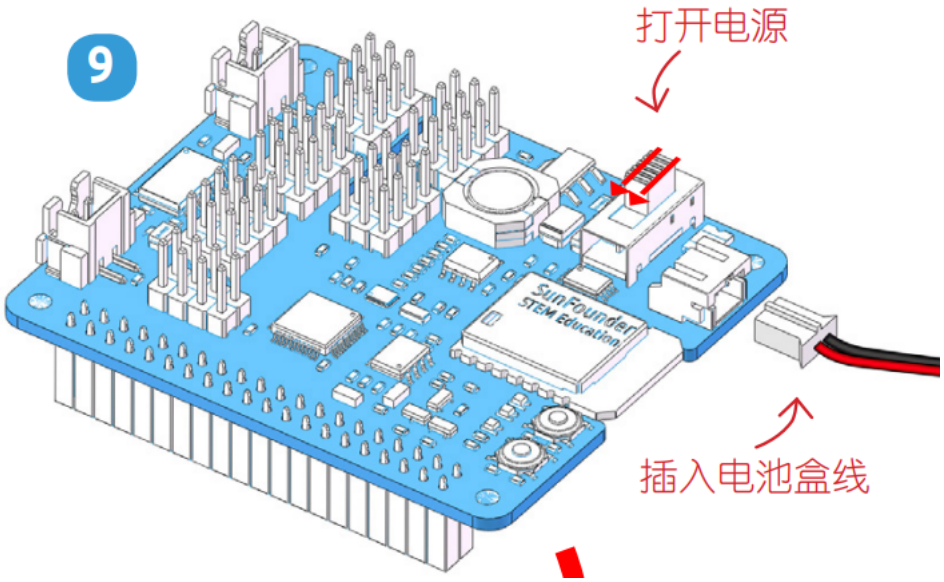
注解： 如果重启后没有声音，可能需要多次运行 i2samp.sh 脚本。

3.1.5 舵机调零

1. 为确保舵机已正确设置为 0°，首先将摇臂轻轻插入舵机轴，然后将摇臂轻轻旋转到不同的角度。



2. 按照组装折页上的提示，插入电池盒线，将电源开关拨向 ON 的位置。等待 1-2 分钟，会有声音提示树莓派启动成功。

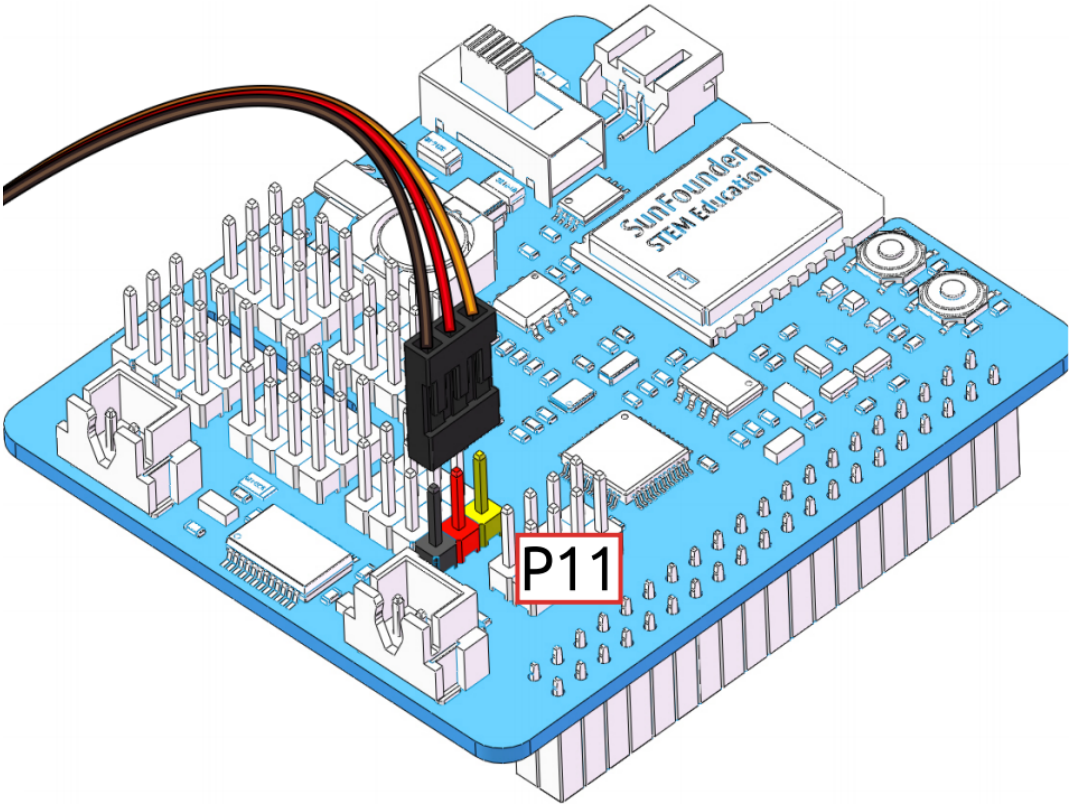


3. 现在，运行 `examples/` 文件夹中的 `servo_zeroing.py`.

```
cd /home/pi/picrawler/examples
sudo python3 servo_zeroing.py
```

注解： 如果报错，请尝试重新启用树莓派的的 I2C 端口，请参阅：*I2C 配置*.

4. 接下来，将舵机线插入 P11 端口，如下所示：



5. 此时你会看到舵机臂转动到特定的位置（0°）。如果伺服臂没有返回到 0°，请按 **RST** 按钮重新启动 Robot HAT。
6. 现在你就可以按照组装折页上的指示继续安装。

注解:

- 在用舵机螺丝固定该舵机前，不要拔出该舵机线，可在固定完之后拔出。
- 不要在上电情况下随意转动舵机以免损坏；如果舵机轴插入的角度不对，需把舵机拔出再重新插入。
- 在组装每个舵机之前，都需要将舵机电缆插入 P11 并打开电源以将它的角度设置为 0°。

3.2 校准 PiCrawler

因为 PiCrawler 安装过程中可能存在偏差或舵机本身的限制，使一些舵机角度略有倾斜，因此您可以对其进行校准。

当然如果你认为组装很完美，不需要校准，你也可以跳过这一章。

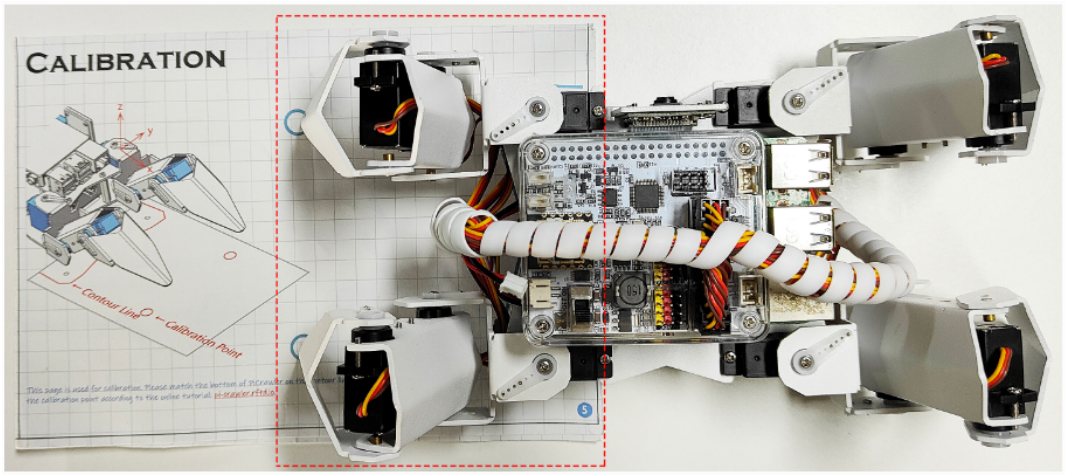
```
cd /home/pi/picrawler/examples/calibration
sudo python3 calibration.py
```

运行上面的代码之后，你会看见终端显示如下界面。

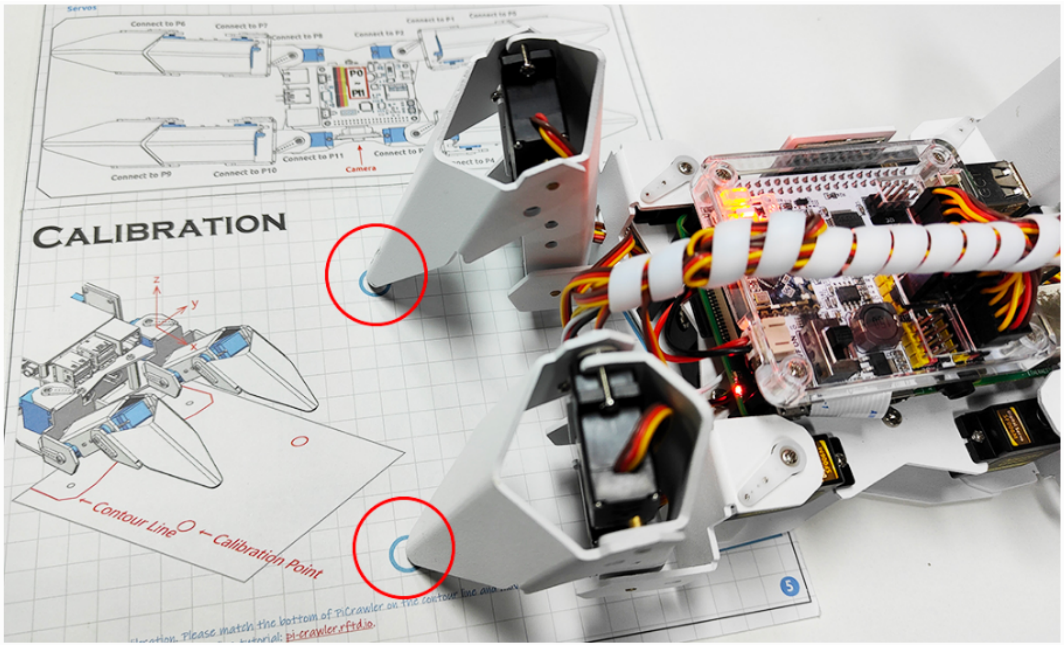
- 按下 1234 来分别选择脚，`1`：右前脚，`2`：左前脚，`3`：左后脚，`4`：右后脚
- 按下 `w`，`a`，`s`，`d`，`r`，和 `f` 来慢慢控制 PiCrawler 的坐标值。
- 按空格保存校准。

具体的步骤如下：

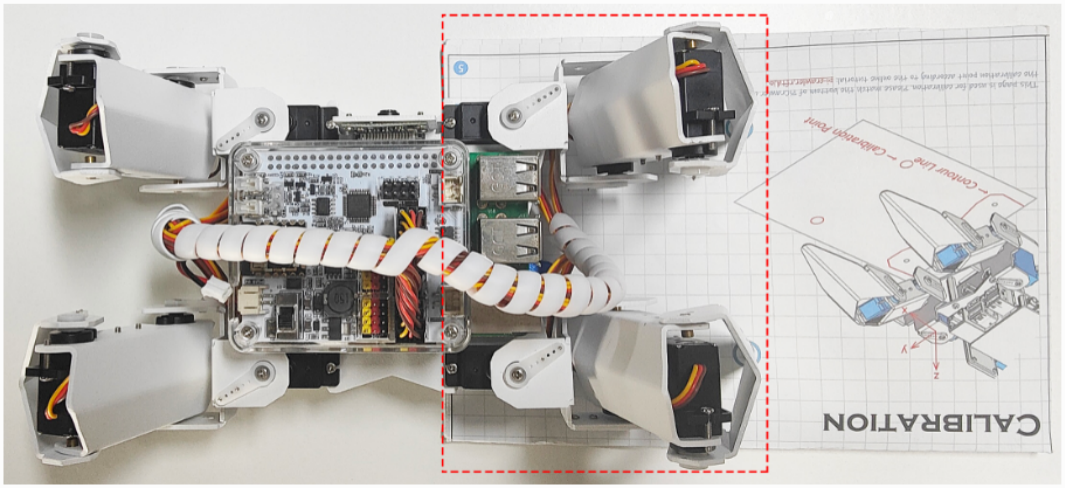
1. 拿出组装折页，将它翻到最后一页，平整的放桌上。然后将 PiCrawler 如下图放置，需要将它的底部与校准图上的轮廓线对齐。



2. 先选择 2 和 3 将左侧两条腿移动到校准点。



3. 然后将校准纸换到右边，选择 1 和 4 将右侧 2 条腿移动到校准点。

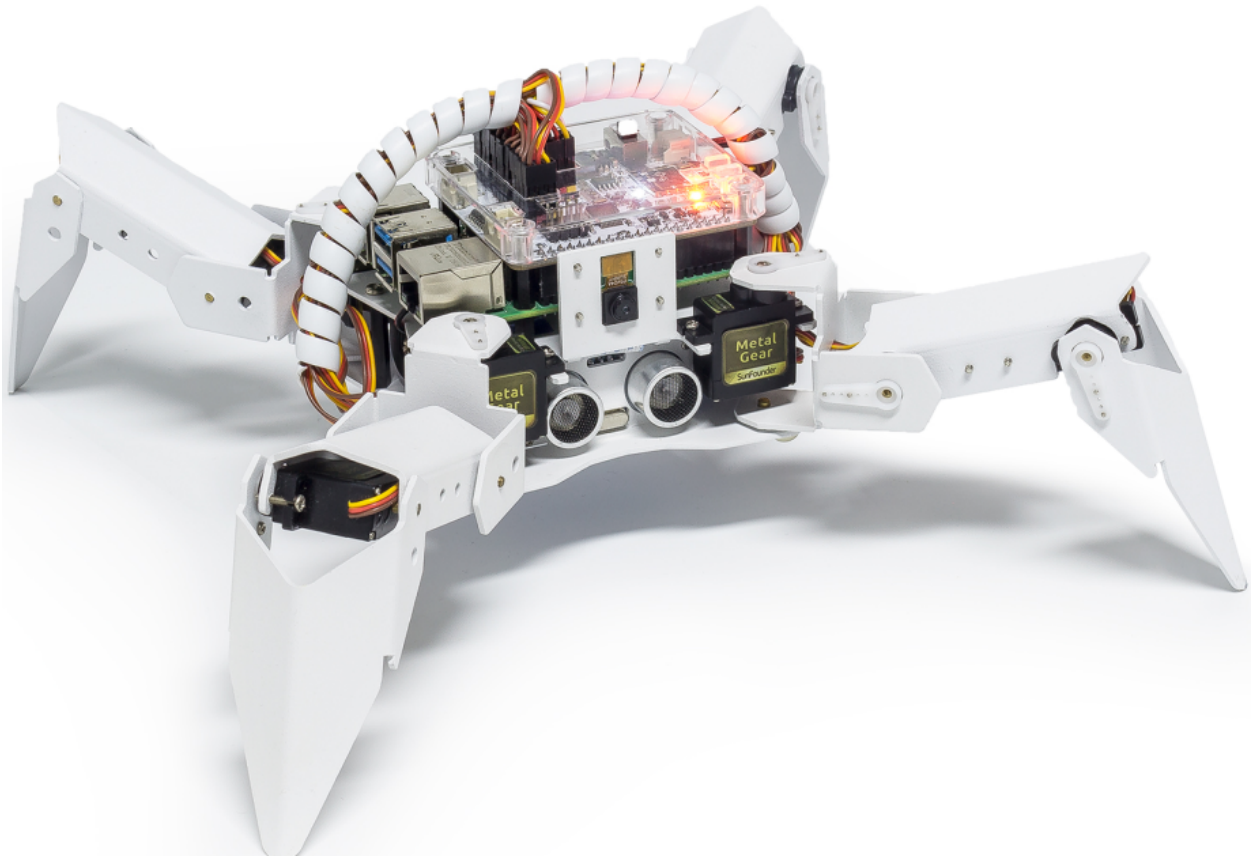


4. 校准完毕后按 空格键进行保存，会提示输入 `Y` 确认，然后 `ctrl+c` 退出程序完成校准。

组装完成后，您可以尝试运行下面的项目。

3.3 移动

这是 PiCrawler's 的第一个项目，让它移动起来。



运行代码

```
cd /home/pi/picrawler/examples
sudo python3 move.py
```

执行代码之后，PiCrawler 会依次执行以下动作：前进、后退、左转、右转、站立。

代码

```
from picrawler import Picrawler
from time import sleep

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

def main():

    speed = 100

    while True:

        crawler.do_action('forward',2,speed)
        sleep(0.05)
        crawler.do_action('backward',2,speed)
        sleep(0.05)
        crawler.do_action('turn left',2,speed)
        sleep(0.05)
        crawler.do_action('turn right',2,speed)
```

(下页继续)

(续上页)

```
        sleep(0.05)
        crawler.do_action('turn left angle',2,speed)
        sleep(0.05)
        crawler.do_action('turn right angle',2,speed)
        sleep(0.05)
        crawler.do_step('stand',speed)
        sleep(1)

if __name__ == "__main__":
    main()
```

这个怎么运作?

首先从您安装的 `Picrawler` 库中导入 `picrawler` 类, 这个类包含了 PiCrawler 的所有操作和实现它们的函数。

```
from picrawler import Picrawler
```

然后示例化 `Picrawler` 类，创建一个它的对象 `crawler`.

```
crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
```

最后用 `crawler.do_action()` 函数来使 Picrawler 移动。

```
crawler.do_action('forward',2,speed)
crawler.do_action('backward',2,speed)
crawler.do_action('turn left',2,speed)
crawler.do_action('turn right',2,speed)
crawler.do_action('turn left angle',2,speed)
crawler.do_action('turn right angle',2,speed)
```

一般情况下，PiCrawler 的所有动作都可以通过 `do_action()` 函数来实现。它有 3 个参数:

- `motion_name` 是具体动作的名字, 包括: `forward`, `turn right`, `turn left`, `backward`, `turn left angle`, `turn right angle`.
- `step` 表示每个动作执行的次数，默认为 1 次。
- `speed` 表示动作执行的速度，默认为 50，范围为 0~100。

此外, `crawler.do_step('stand',speed)` 在这里也是用于使 PiCrawler 站立。它的用法将在下面的例子中说明。

3.4 键盘控制

在这个项目中，我们将学习如何使用键盘远程控制 PiCrawler。您可以控制 PiCrawler 向前、向后、向左和向右移动。

运行代码

```
cd /home/pi/picrawler/examples
sudo python3 keyboard_control.py
```

代码运行后，请根据终端弹出的提示进行操作。

- 按下 `w`，让 PiCrawler 前进。

- 按下 a，让 PiCrawler 左转。
- 按下 s，让 PiCrawler 后退。
- 按下 d，让 PiCrawler 右转。
- 按下 esc 退出程序。

代码

```
from picrawler import Picrawler
from time import sleep
import readchar

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
speed = 90

manual = '''
Press keys on keyboard to control PiCrawler!
    w: Forward
    a: Turn left
    s: Backward
    d: Turn right
    esc: Quit
'''

def show_info():
    print("\033[H\033[J",end='')  # clear terminal windows
    print(manual)


def main():
    show_info()
    while True:
        key = readchar.readkey()
        key = key.lower()
        if key in('wsad'):
            if 'w' == key:
                crawler.do_action('forward',1,speed)
            elif 's' == key:
                crawler.do_action('backward',1,speed)
            elif 'a' == key:
                crawler.do_action('turn left',1,speed)
            elif 'd' == key:
                crawler.do_action('turn right',1,speed)
            sleep(0.05)
            show_info()
        elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
            print("\n Quit")
            break

        sleep(0.02)


if __name__ == "__main__":
    main()
```

这个怎么运作?

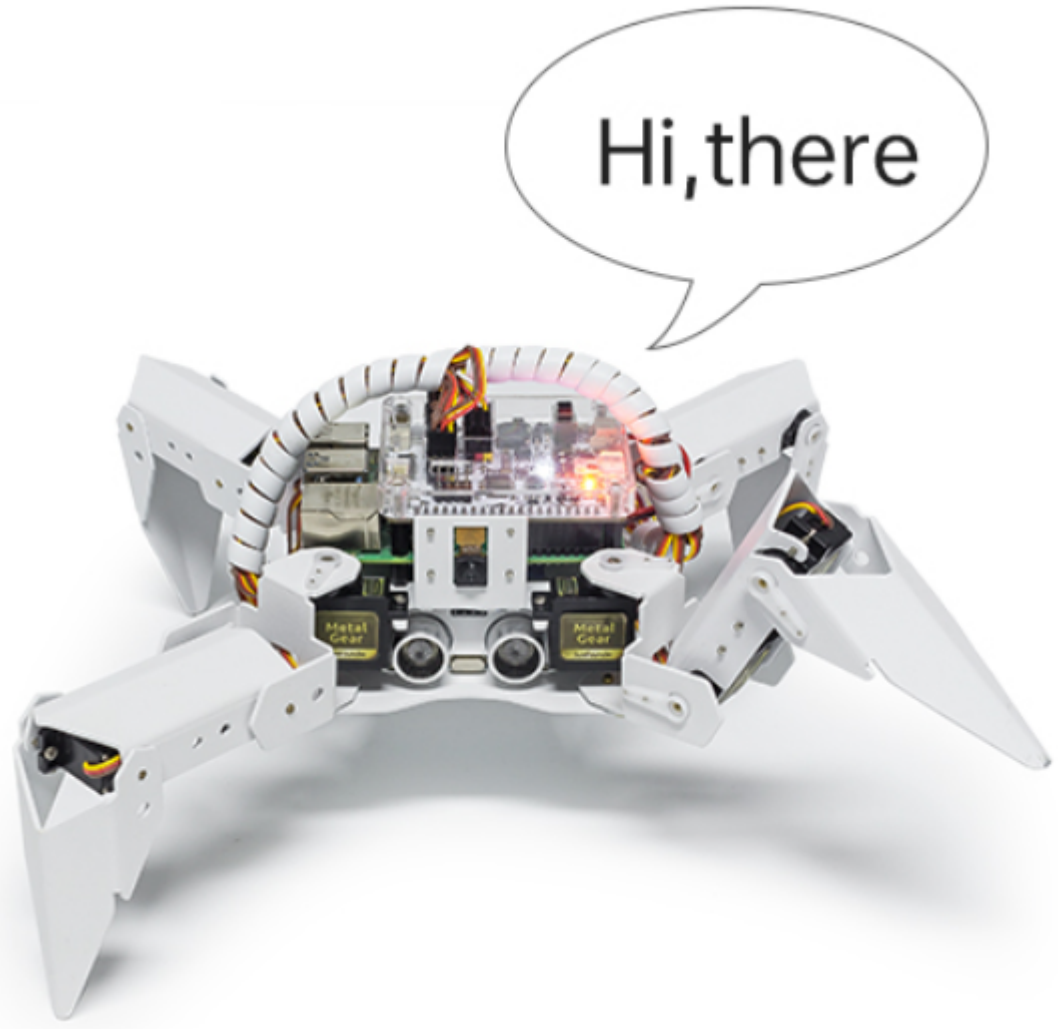
根据读取的键盘字符，让 PiCrawler 做我们设置的动作。`lower()` 是将读取的按键字符转化成小写字符，这

样无论读取了该字母的大小写，都是有效的。

```
while True:
    key = readchar.readkey()
    key = key.lower()
    if key in('wsad'):
    if 'w' == key:
        crawler.do_action('forward',1,speed)
    elif 's' == key:
        crawler.do_action('backward',1,speed)
    elif 'a' == key:
        crawler.do_action('turn left',1,speed)
    elif 'd' == key:
        crawler.do_action('turn right',1,speed)
    sleep(0.05)
    show_info()
    elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
    print("\n Quit")
        break
```

3.5 音效

在本例中，我们使用 PiCrawler（准确地说是 Robot HAT）的音效。它由三部分组成，分别是 **Muisc**, **Sound**, **Text to Speech**.



安装 i2samp

在使用该功能之前，请先激活扬声器，使其启用并可以发出声音。

运行 `i2samp.sh` ，此脚本将安装使用 i2s 放大器所需的一切依赖。

```
cd /home/pi/picrawler/
sudo bash i2samp.sh
```

过程中会有提示请求确认，全部输入 **Y** 即可。对 Raspberry Pi 系统进行更改后，需要重新启动计算机才能使这些更改生效。

重新启动后，再次运行脚本 `i2samp.sh` 以测试放大器。如果扬声器成功播放声音，则配置完成。

运行代码

```
cd /home/pi/picrawler/examples
sudo python3 sound_effect.py
```

代码运行后，请按照终端打印的提示进行操作。

- 按下 `q` 来播放背景音乐
- 按下 `1` 来播放音效

- 按下 2 来以单独的线程播放音效
- 按下 t 来让 PiCrawler 说 Hello。

代码

```
from time import sleep
from robot_hat import Music,TTS
import readchar

music = Music()
tts = TTS()

manual = '''
Input key to call the function!
    q: Play background music
    1: Play sound effect
    2: Play sound effect with threads
    t: Text to speak
'''

def main():
    print(manual)

    flag_bgm = False
    music.music_set_volume(20)
    tts.lang("en-US")


    while True:
        key = readchar.readkey()
        key = key.lower()
        if key == "q":
            flag_bgm = not flag_bgm
            if flag_bgm is True:
                music.background_music('./musics/sports-Ahjay_Stelino.mp3')
            else:
                music.music_stop()

        elif key == "1":
            music.sound_effect_play('./sounds/talk1.wav')
            sleep(0.05)
            music.sound_effect_play('./sounds/talk3.wav')
            sleep(0.05)
            music.sound_effect_play('./sounds/sign.wav')
            sleep(0.5)

        elif key =="2":
            music.sound_effect_threading('./sounds/talk1.wav')
            sleep(0.05)
            music.sound_effect_threading('./sounds/talk3.wav')
            sleep(0.05)
            music.sound_effect_threading('./sounds/sign.wav')
            sleep(0.5)

        elif key == "t":
            words = "Hello"
            tts.say(words)
```

(下页继续)

(续上页)

```
if __name__ == "__main__":
    main()
```

这个怎么运作?

与背景音乐相关的功能包括:

- `music = Music()`：声明对象。
- `music.music_set_volume(20)`：设置音量，范围为 0~100。
- `music.background_music(./musics/sports-Ahjay_Stelino.mp3)`：播放音乐文件, 参数为文件所在路径，比如 `./musics` 路径下的 **sports-Ahjay_Stelino.mp3** 文件。
- `music.music_stop()`：停止播放背景音乐。

注解: 你可以通过*Filezilla 软件* 给 `musics` or `sounds` 文件夹添加不同的音乐或者音效。

与音效相关的功能包括:

- `music = Music()`
- `music.sound_effect_play('./sounds/talk1.wav')`：播放音效文件, 参数为文件所在路径，比如 `./sounds` 路径下的 **talk1.wav** 文件。
- `muisc.sound_effect_threading('./sounds/talk1.wav')`：开辟一个新的线程来播放音效文件，无需挂起主线程。

与文本到语音相关的功能包括:

- `tts = TTS()`
- `tts.say(words)`：文字音频。
- `tts.lang("en-US")`：设置语言。

注解: 用 `lang("")` 函数来切换各国语言，参数为下列的字符。

表 1: 语言

zh-CN	普通话 (中文)
en-US	英语-美
en-GB	英语-英
de-DE	德语
es-ES	西班牙语
fr-FR	法语
it-IT	意大利语

3.6 避障

在这个项目中，picrawler 将使用超声波模块来检测前方的障碍物。当 PiCrawler 检测到障碍物时，它会发出信号并寻找另一个方向前进。

运行代码

```
cd /home/pi/picrawler/examples
sudo python3 avoid.py
```

代码运行后，PiCrawler 会向前走。如果检测到前方障碍物的距离小于 10cm，就会停下并发出警告，然后左转并继续检测。如果左转之后没有障碍物或障碍物距离大于 10，则继续向前移动。

代码

```
from picrawler import Picrawler
from robot_hat import TTS, Music
from robot_hat import Ultrasonic
from robot_hat import Pin
import time
import os

tts = TTS()
music = Music()
```

(下页继续)

(续上页)

```
crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])
sonar = Ultrasonic(Pin("D2") ,Pin("D3"))

alert_distance = 15

speed = 100

def main():
    distance = sonar.read()
    print(distance)
    if distance < 0:
        pass
    elif distance <= alert_distance:
        try:
            music.sound_effect_threading('./sounds/sign.wav')
        except Exception as e:
            print(e)
        crawler.do_action('turn left angle',3,speed)
        time.sleep(0.2)
    else :
        crawler.do_action('forward', 1,speed)
        time.sleep(0.2)


if __name__ == "__main__":
    while True:
        main()
```

这个怎么运作?

您可以通过导入 `Ultrasonic` 类来帮助你检测障碍物。

```
from robot_hat import Ultrasonic
```

然后初始化超声波引脚。

```
sonar = Ultrasonic(Pin("D2") ,Pin("D3"))
```

如下是主程序的逻辑。

- 读取超声波模块检测到的 `distance` 值，过滤掉小于 0 的值（当超声波模块距离障碍物太远或无法正确读取数据时，会出现 `distance<0` 的情况）。
- 当 `distance` 小于等于 `alert_distance`（之前设置的阈值，数值为 10）时，播放音效 `sign.wav`。并让 PiCrawler 执行 `turn left angle` 动作。
- 当 `distance` 大于 `alert_distance` 时，让 PiCrawler 执行 `forward` 动作。

```
distance = sonar.read()
print(distance)
if distance < 0:
    pass
elif distance <= alert_distance:
    try:
        music.sound_effect_threading('./sounds/sign.wav')
    except Exception as e:
```

(下页继续)

(续上页)

```
        print(e)
    crawler.do_action('turn left angle',3,speed)
    time.sleep(0.2)
else :
    crawler.do_action('forward', 1,speed)
    time.sleep(0.2)
```

注解: 你可以通过*Filezilla 软件* 给 musics or sounds 文件夹添加不同的音乐或者音效。

3.7 计算机视觉

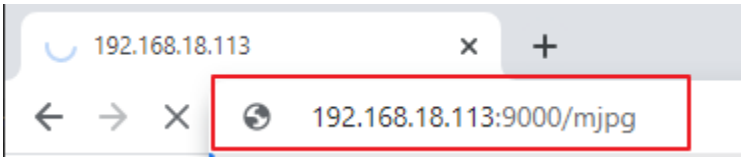
本项目将正式进入计算机视觉领域!

运行代码

```
cd /home/pi/picrawler/examples
sudo python3 display.py
```

查看图像

代码运行后，在浏览器中输入 http://<your IP>:9000/mjpg 来查看视频画面。如: https://192.168.18.113:9000/mjpg



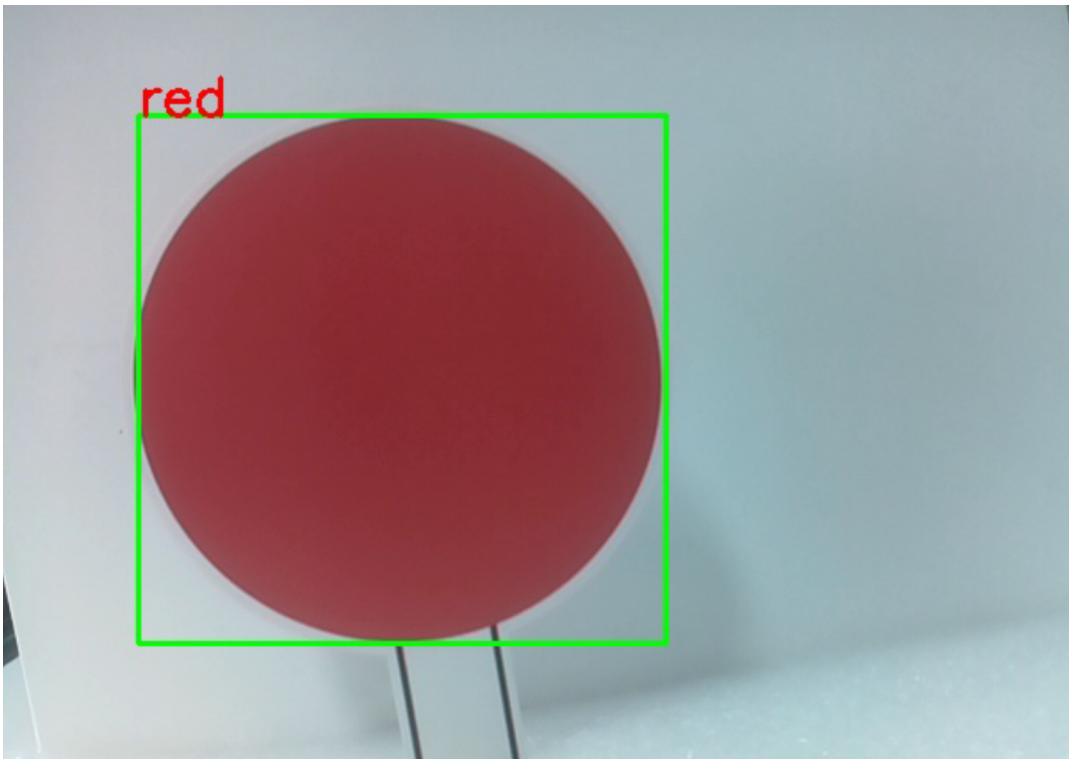
按照根据终端的信息提示，按下键盘上的按键来查看相应的功能。

- q: 拍照
- 1: 颜色检测：红色: red
- 2: 颜色检测：橙色色: orange
- 3: 颜色检测：黄色: yellow
- 4: 颜色检测：绿色: green
- 5: 颜色检测：蓝色: blue
- 6: 颜色检测：紫色: purple
- 0: 关闭颜色检测
- r：扫描二维码
- f: 开/关人脸检测
- s: 显示检测的目标信息
- **拍照**

  在终端中输入 q 并按下回车。相机当前看到的图片会被保存（如果开启颜色检测功能，保存的图片中也会出现标记框）。你可以从目录 /home/pi/Pictures/PiCrawler/ 看到这些照片。然后用*Filezilla 软件* 之类的工具将照片发送到你的电脑上。

- **颜色检测**

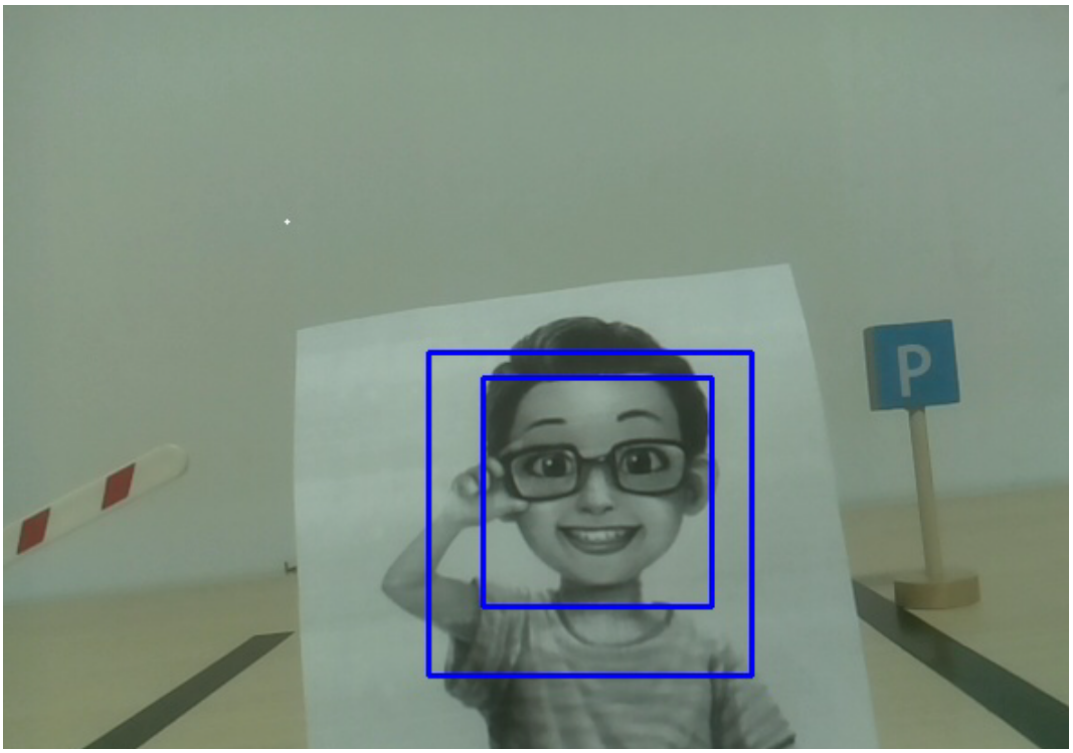
  在 1~6 之间输入一个数字，将会检测”红色、橙色、黄色、绿色、蓝色、紫色”中的一种颜色。输入 0 以关闭颜色检测。



注解： 你可以下载并打印 PDF 颜色卡片来用于颜色检测。

- **人脸检测**

  输入 f 来打开人脸检测。



- **二维码检测**

  输入 `r` 来打开二维码检测。在识别二维码之前，不能进行其他操作。二维码的解码信息将打印在终端中。

- **显示信息**

输入 s 会在终端打印人脸检测（和颜色检测）目标的信息。包括被测物体的中心坐标（X，Y）和尺寸（重量，高度）。

代码

```
from pydoc import text
from vilib import Vilib
from time import sleep, time, strftime, localtime
import threading
import readchar


flag_face = False
flag_color = False
qr_code_flag = False

manual = '''
Input key to call the function!
    q: Take photo
    1: Color detect : red
    2: Color detect : orange
    3: Color detect : yellow
    4: Color detect : green
    5: Color detect : blue
    6: Color detect : purple
    0: Switch off Color detect
    r: Scan the QR code
    f: Switch ON/OFF face detect
    s: Display detected object information
'''

color_list = ['close', 'red', 'orange', 'yellow',
        'green', 'blue', 'purple',
]

def face_detect(flag):
    print("Face Detect:" + str(flag))
    Vilib.face_detect_switch(flag)


def qrcode_detect():
    global qr_code_flag
    if qr_code_flag == True:
        Vilib.qrcode_detect_switch(True)
        print("Waitting for QR code")

    text = None
    while True:
        temp = Vilib.detect_obj_parameter['qr_data']
        if temp != "None" and temp != text:
            text = temp
            print('QR code:%s'%text)
        if qr_code_flag == False:
            break
        sleep(0.5)
    Vilib.qrcode_detect_switch(False)
```

(下页继续)

(续上页)

```
def take_photo():
    _time = strftime('%Y-%m-%d-%H-%M-%S',localtime(time()))
    name = 'photo_%s'%_time
    path = "/home/pi/Pictures/PiCrawler/"
    Vilib.take_photo(name, path)
    print('photo save as %s%s.jpg'%(path,name))


def object_show():
    global flag_color, flag_face

    if flag_color is True:
        if Vilib.detect_obj_parameter['color_n'] == 0:
            print('Color Detect: None')
        else:
            color_coodinate = (Vilib.detect_obj_parameter['color_x'],Vilib.detect_obj_
↪parameter['color_y'])
            color_size = (Vilib.detect_obj_parameter['color_w'],Vilib.detect_obj_
↪parameter['color_h'])
            print("[Color Detect] ","Coordinate:",color_coodinate,"Size",color_size)

    if flag_face is True:
        if Vilib.detect_obj_parameter['human_n'] == 0:
            print('Face Detect: None')
        else:
            human_coodinate = (Vilib.detect_obj_parameter['human_x'],Vilib.detect_obj_
↪parameter['human_y'])
            human_size = (Vilib.detect_obj_parameter['human_w'],Vilib.detect_obj_
↪parameter['human_h'])
            print("[Face Detect] ","Coordinate:",human_coodinate,"Size",human_size)


def main():
    global flag_face, flag_color, qr_code_flag
    qrcode_thread = None

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    print(manual)

    while True:
        # readkey
        key = readchar.readkey()
        key = key.lower()
        # take photo
        if key == 'q':
            take_photo()
        # color detect
        elif key != '' and key in ('0123456'):  # '' in ('0123') -> True
            index = int(key)
            if index == 0:
                flag_color = False
                Vilib.color_detect('close')
            else:
                flag_color = True
                Vilib.color_detect(color_list[index]) # color_detect(color:str ->␣
↪color_name/close)
```

(下页继续)

(续上页)

```
            print('Color detect : %s'%color_list[index])
        # face detection
        elif key =="f":
            flag_face = not flag_face
            face_detect(flag_face)
        # qrcode detection
        elif key =="r":
            qr_code_flag = not qr_code_flag
            if qr_code_flag == True:
                if qrcode_thread == None or not qrcode_thread.is_alive():
                    qrcode_thread = threading.Thread(target=qrcode_detect)
                    qrcode_thread.setDaemon(True)
                    qrcode_thread.start()
            else:
                if qrcode_thread != None and qrcode_thread.is_alive():
                # wait for thread to end
                    qrcode_thread.join()
                    print('QRcode Detect: close')
        # show detected object information
        elif key == "s":
            object_show()

        sleep(0.5)


if __name__ == "__main__":
    main()
```

这个怎么运作?

这里首先需要注意的是下面的函数。这两个函数可以帮助您启动相机。

```
Vilib.camera_start()
Vilib.display()
```

与" 物体检测" 相关的函数:

- `Vilib.face_detect_switch(True)`：开启/关闭人脸检测
- `Vilib.color_detect(color)`：对于颜色检测，只能同时检测一种颜色。可以输入的参数有: `"red"`, `"orange"`, `"yellow"`, `"green"`, `"blue"`, `"purple"`
- `Vilib.color_detect_switch(False)`：关闭颜色检测
- `Vilib.qrcode_detect_switch(False)`：开启/关闭二维码检测，返回二维码的解码数据
- `Vilib.gesture_detect_switch(False)`：打开/关闭手势检测
- `Vilib.traffic_sign_detect_switch(False)`：开启/关闭交通标志检测

目标检测到的信息将存储在 `detect_obj_parameter = Manager().dict()` 字典中。

在主程序中，您可以像这样使用它:

```
Vilib.detect_obj_parameter['color_x']
```

字典的键及其用途如下表所示:

- `color_x`: 检测到的色块中心坐标的 x 值，范围 0~320
- `color_y`: 检测到的色块中心坐标的 y 值，范围 0~240

- color_w：检测色块的宽度，范围 0~320
- color_h：检测到的色块高度，范围 0~240
- color_n: 检测到的色块数量
- human_x：检测到的人脸中心坐标的 x 值，范围 0~320
- human_y：检测人脸中心坐标的 y 值，范围 0~240
- human_w：检测到的人脸宽度，范围 0~320
- human_h：检测到的人脸高度，范围 0~240
- human_n：检测到的人脸数量
- traffic_sign_x：检测到的交通标志的中心坐标 x 值，范围 0~320
- traffic_sign_y：检测到的交通标志的中心坐标 y 值，范围 0~240
- traffic_sign_w：检测到的交通标志的宽度，范围 0~320
- traffic_sign_h：检测到的交通标志的高度，范围 0~240
- traffic_sign_t: 检测到的交通标志的内容，取值列表为 *['stop','right','left','forward']*
- gesture_x：检测到的手势的中心坐标 x 值，范围 0~320
- gesture_y：检测到的手势的中心坐标 y 值，范围 0~240
- gesture_w：检测到的手势宽度，范围 0~320
- gesture_h：检测到的手势高度，范围 0~240
- gesture_t：检测到的手势内容，值列表为 *["paper","scissor","rock"]*
- qr_date: 正在检测的二维码内容
- qr_x：待检测二维码的中心坐标 x 值，范围 0~320
- qr_y：待检测二维码的中心坐标 y 值，范围 0~240
- qr_w：要检测的二维码宽度，范围 0~320
- qr_h：要检测的二维码高度，范围 0~320

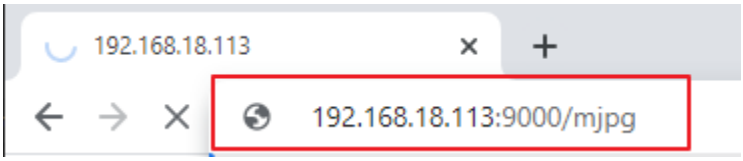
3.8 录制视频

本示例将指导您如何使用录制视频功能。

运行代码

```
cd /home/pi/picrawler/examples
sudo python3 record_video.py
```

代码运行后，在浏览器中输入 http://<your IP>:9000/mjpg 来查看视频画面。比如: https://192.168.18.113:9000/mjpg



然后按下键盘上的按键来录像或停止录像。

- 按下 q 键开始/暂停/继续录像，按下 e 停止录像并保存文件。
- 按下 esc 退出程序。

代码

```
from time import sleep,strftime,localtime
from vilib import Vilib
import readchar

manual = '''
Press keys on keyboard to control recording:
        Q: record/pause/continue
        E: stop
        ESC: Quit
'''

def print_overwrite(msg,  end='', flush=True):
        print('\r\033[2K', end='',flush=True)
        print(msg, end=end, flush=True)

def main():
        rec_flag = 'stop' # start,pause,stop
        vname = None
        Vilib.rec_video_set["path"] = "/home/pi/Videos/" # set path

        Vilib.camera_start(vflip=False,hflip=False)
        Vilib.display(local=True,web=True)
        sleep(0.8)  # wait for startup

        print(manual)
        while True:
                # read keyboard
                key = readchar.readkey()
                key = key.lower()
                # start,pause
                if key == 'q':
                        key = None
                        if rec_flag == 'stop':
                                rec_flag = 'start'
                                # set name
                                vname = strftime("%Y-%m-%d-%H.%M.%S", localtime())
                                Vilib.rec_video_set["name"] = vname
                                # start record
                                Vilib.rec_video_run()
                                print_overwrite('rec start ...')
                        elif rec_flag == 'start':
                                rec_flag = 'pause'
                                Vilib.rec_video_pause()
                                print_overwrite('pause')
                        elif rec_flag == 'pause':
                                rec_flag = 'start'
                                Vilib.rec_video_start()
                                print_overwrite('continue')
                # stop
                elif key == 'e' and rec_flag != 'stop':
                        key = None
                        rec_flag = 'stop'
                        Vilib.rec_video_stop()
```

(下页继续)

(续上页)

```
                        print_overwrite("The video saved as %s%s.avi"%(Vilib.rec_
↪video_set["path"],vname),end='\n')
                # quit
                elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_
↪SEQUENCES:
                        Vilib.camera_close()
                        print('\nquit')
                        break

                sleep(0.1)

if __name__ == "__main__":
        main()
```

这个怎么运作?

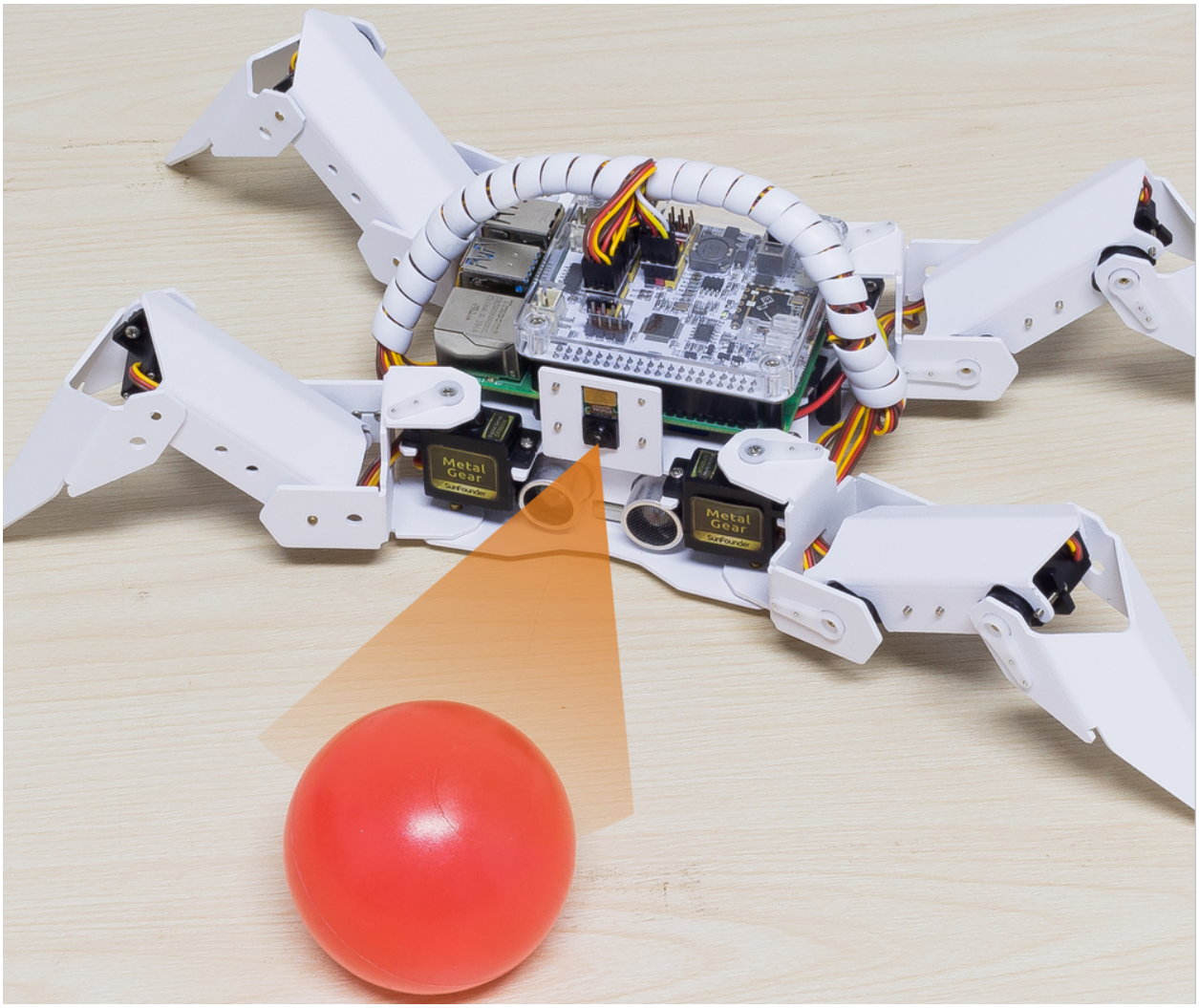
与录像相关的功能包括:

- `Vilib.rec_video_run(video_name)`：启动线程来录制视频。`video_name` 是一个字符串，表示视频文件的名称。
- `Vilib.rec_video_start()`：开始或继续视频录制。
- `Vilib.rec_video_pause()`：暂停录像。
- `Vilib.rec_video_stop()`：停止录像。

`Vilib.rec_video_set["path"] = "/home/pi/video/test/"` 设置视频文件的存储位置。

3.9 斗牛

让 PiCrawler 成为愤怒的公牛！利用它的相机追踪功能来冲撞红布！



运行代码

```
cd /home/pi/picrawler/examples
sudo python3 bull_fight.py
```

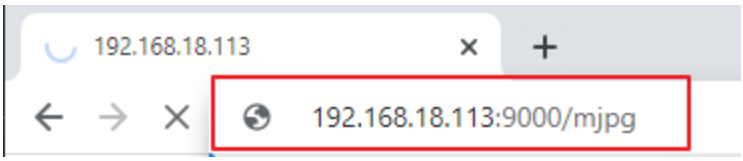
查看图像

代码运行后，如果你在 PiCrawler 前面放一个红色的物体，比如红布，红球，红色卡片之类的，它就会随着这个红色的物体移动。

同时终端会显示如下提示:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

代表你可以在浏览器中输入 `http://<your IP>:9000/mjpg` 来查看视频画面。比如: `https://192.168.18.113:9000/mjpg`



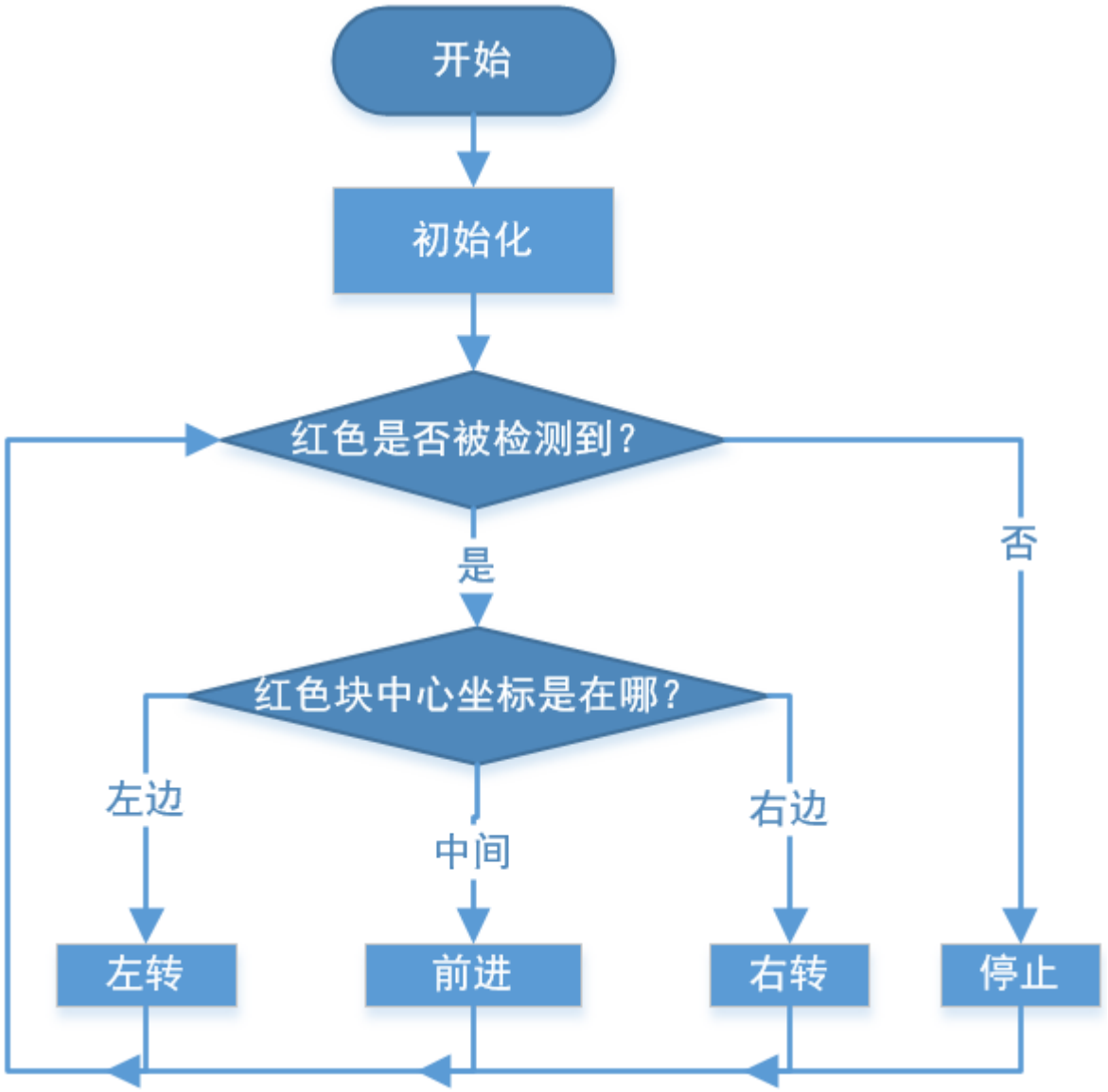
代码

```
from picrawler import Picrawler
from time import sleep
from robot_hat import Music
from vilib import Vilib


crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

music = Music()

def main():
    Vilib.camera_start()
    Vilib.display()
    Vilib.color_detect("red")
    speed = 100

    while True:
        if Vilib.detect_obj_parameter['color_n']!=0:
            coordinate_x = Vilib.detect_obj_parameter['color_x']
            music.sound_effect_threading('./sounds/talk1.wav')

            if coordinate_x < 100:
                crawler.do_action('turn left',1,speed)
                sleep(0.05)
            elif coordinate_x > 220:
                crawler.do_action('turn right',1,speed)
                sleep(0.05)
            else :
                crawler.do_action('forward',2,speed)
                sleep(0.05)
        else :
            crawler.do_step('stand',speed)
            sleep(0.05)


if __name__ == "__main__":
    main()
```

这个怎么运作?

总的来说，这个项目结合了移动, 计算机视觉 和音效 的知识点。

其流程如下图所示:



3.10 寻宝

在你的房间里布置一个迷宫，在六个角落放置六张不同颜色的卡片。然后控制 PiCrawler 一一搜索这些色卡吧！

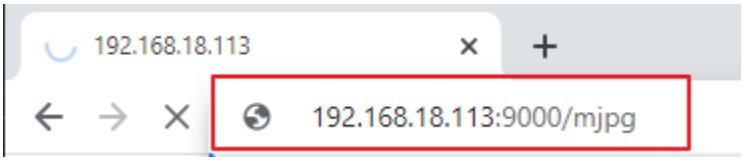
注解： 您可以下载并打印文件 PDF 颜色卡片用于颜色检测。

运行代码

```
cd /home/pi/picrawler/examples
sudo python3 treasure_hunt.py
```

查看图像

代码运行后，在浏览器中进入 `http://<your IP>:9000/mjpg` 来查看视频画面。比如: `https://192.168.18.113:9000/mjpg`



然后按照终端打印的提示进行操作。

- 按下 `w`，让 PiCrawler 前进。
- 按下 `a`，让 PiCrawler 左转。
- 按下 `s`，让 PiCrawler 后退。
- 按下 `d`，让 PiCrawler 右转。
- 按下 空格键来让 PiCrawler 说一种颜色。
- 按下 `esc` 退出游戏。

代码

```
from picrawler import Picrawler
from time import sleep
from robot_hat import Music,TTS
from vilib import Vilib
import readchar
import random
import threading

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

music = Music()
tts = TTS()

manual = '''
Press keys on keyboard to control Picrawler!
        w: Forward
        a: Turn left
        s: Backward
        d: Turn right
        space: Say the target again
        esc: Quit
'''

color = "red"
color_list=["red","orange","yellow","green","blue","purple"]
key_dict = {
        'w': 'forward',
        's': 'backward',
        'a': 'turn_left',
        'd': 'turn_right',
}
def renew_color_detect():
        global color
        color = random.choice(color_list)
        Vilib.color_detect(color)
```

(下页继续)

(续上页)

```
        tts.say("Look for " + color)

key = None
lock = threading.Lock()
def key_scan_thread():
        global key
        while True:
                key_temp = readchar.readkey()
                print('\r',end='')
                with lock:
                        key = key_temp.lower()
                        if key == readchar.key.SPACE:
                                key = 'space'
                        elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_
↪SEQUENCES:
                                key = 'quit'
                                break
                sleep(0.01)

def main():
        global key
        action = None
        Vilib.camera_start(vflip=False,hflip=False)
        Vilib.display(local=False,web=True)
        sleep(0.8)
        speed = 100
        print(manual)

        sleep(1)
        _key_t = threading.Thread(target=key_scan_thread)
        _key_t.setDaemon(True)
        _key_t.start()

        tts.say("game start")
        sleep(0.05)
        renew_color_detect()
        while True:

                if Vilib.detect_obj_parameter['color_n']!=0 and Vilib.detect_obj_
↪parameter['color_w']>100:
                        tts.say("will done")
                        sleep(0.05)
                        renew_color_detect()

                with lock:
                        if key != None and key in ('wsad'):
                                action = key_dict[str(key)]
                                key =  None
                        elif key == 'space':
                                tts.say("Look for " + color)
                                key =  None
                        elif key == 'quit':
                                _key_t.join()
                                Vilib.camera_close()
                                print("\n\rQuit")
                                break
```

(下页继续)

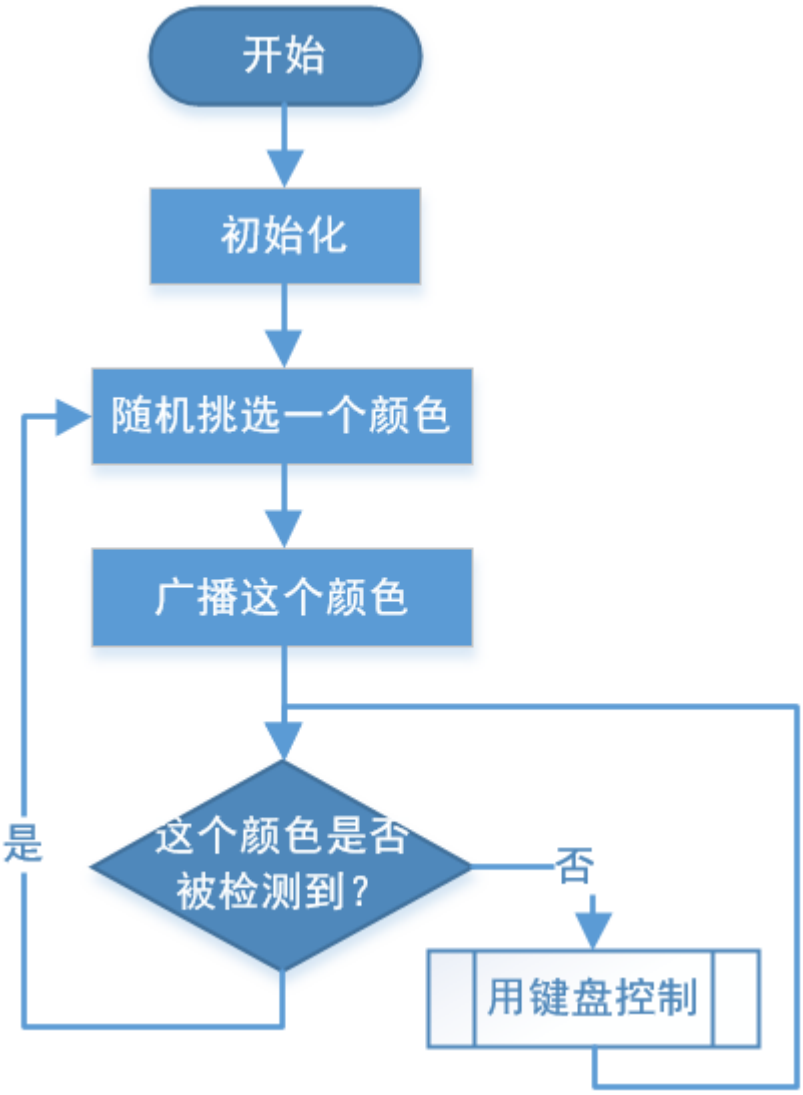
(续上页)

```
                if action != None:
                        crawler.do_action(action,1,speed)
                        action = None

                sleep(0.05)


if __name__ == "__main__":
        main()
```

怎么运行的?

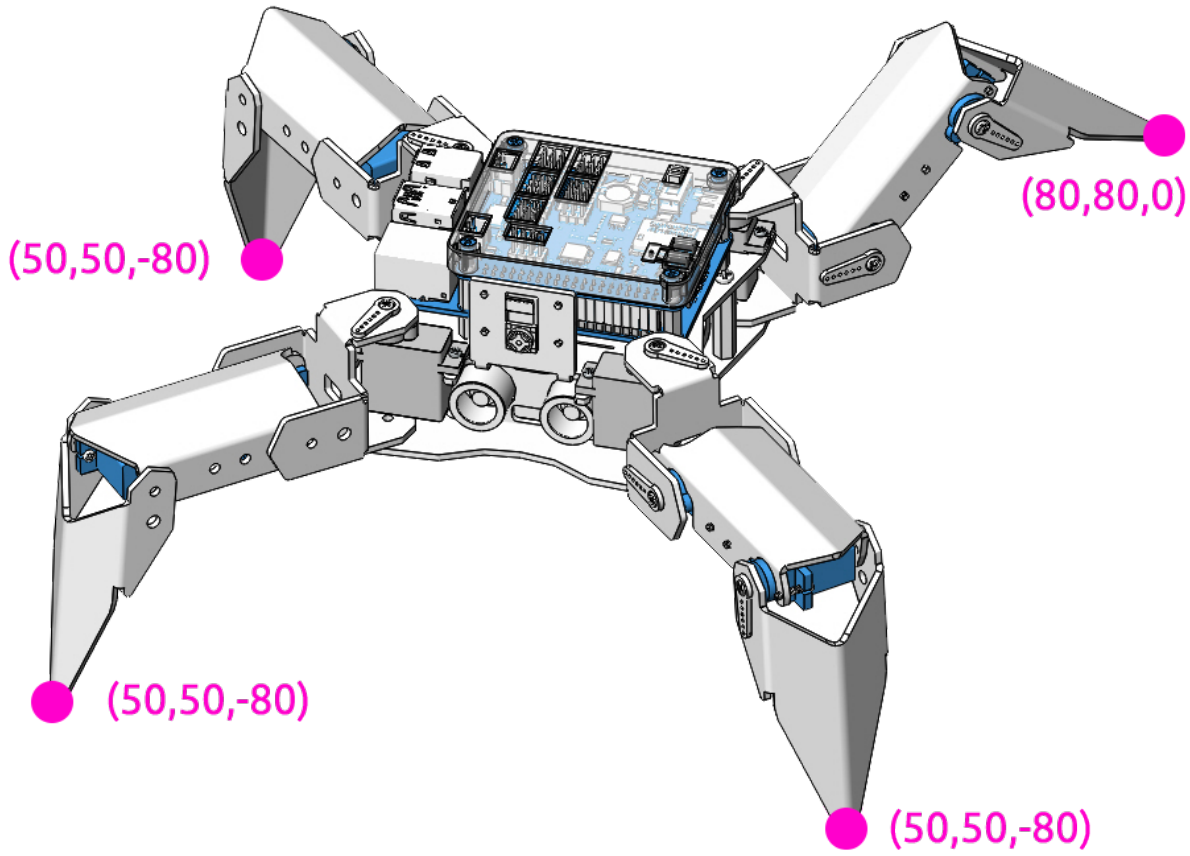
总的来说，这个项目结合了键盘控制, 计算机视觉 和音效。

其流程如下图所示:



3.11 摆姿势

PiCrawler 可以通过写入坐标数组来摆出各种姿势。在这里，它摆出抬起右后脚的姿势。



运行代码

```
cd /home/pi/picrawler/examples
sudo python3 do_step.py
```

代码运行后，你会看到 PiCrawler 抬起右脚。

代码

```
from picrawler import Picrawler
from time import sleep

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

## [right front],[left front],[left rear],[right rear]
new_step=[[50, 50, -80], [50, 50, -80],[80, 80, 0], [50, 50, -80]]

def main():

    speed = 100
```

(下页继续)

(续上页)

```
    while True:

        crawler.do_step('stand',speed)
        print(crawler.step_list.get('stand'))
        sleep(3)
        crawler.do_step(new_step,speed)
        print(new_step)
        sleep(3)

if __name__ == "__main__":
    main()
```

这个怎么运作?

在这段代码中需要注意的是 `crawler.do_step()`。

与 `do_action()` 类似, `do_step()` 也可以操纵 PiCrawler 的行为。不同的是前者用于执行连贯的动作如 `move forward`, 而后者用于执行单独的动作如 `stand` 和 `sit`。

它有两种用法:

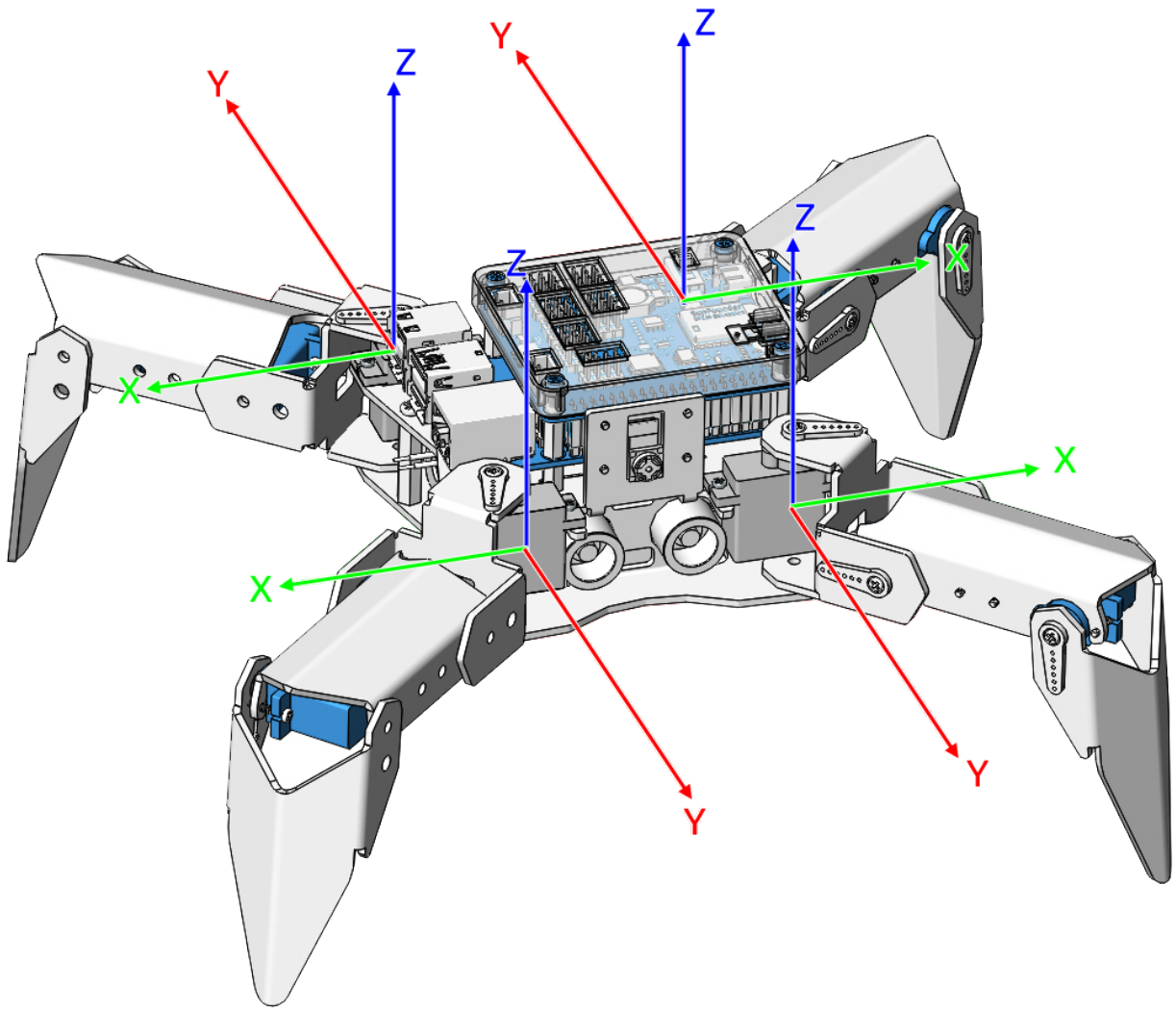
一: 参数可以接收字符串, 直接使用 `picrawler` 库中的 `step_list` 字典。

```
crawler.do_step('stand',speed)
# "speed" indicates the speed of the step, the range is 0~100.
```

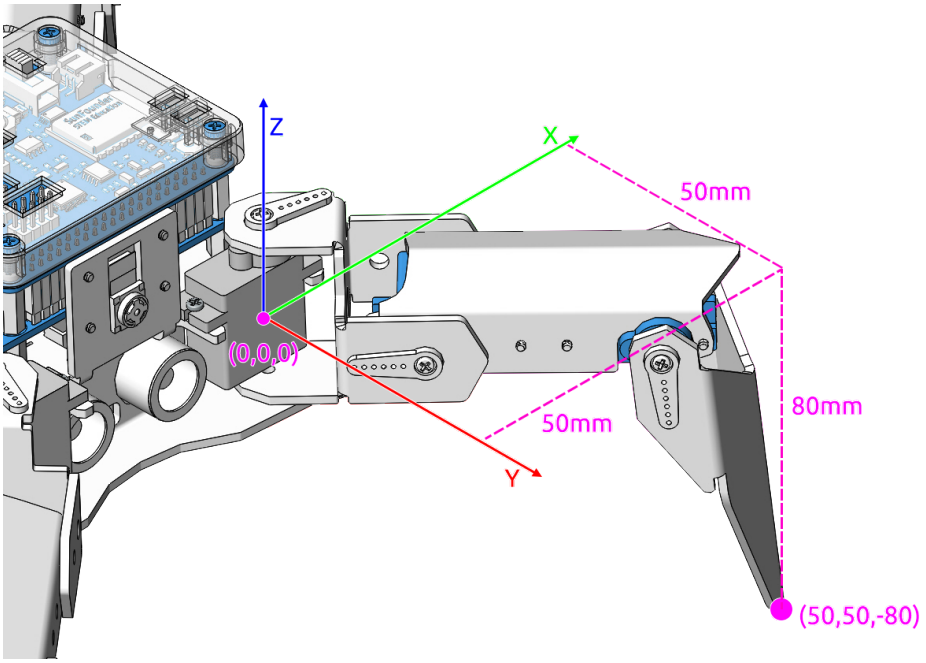
二: 也可以接收一个 4 个坐标值的数组。

```
crawler.do_step([[50, 50, -80], [50, 50, -80],[80, 80, 0], [50, 50, -80]],speed)
# These four coordinates are used to control the four legs of right front, left front,
↪ left rear, and left rear respectively.
```

每只脚都有一个独立的空间直角坐标系。如下所示: :



您需要单独测量每个脚趾的坐标。如下所示：



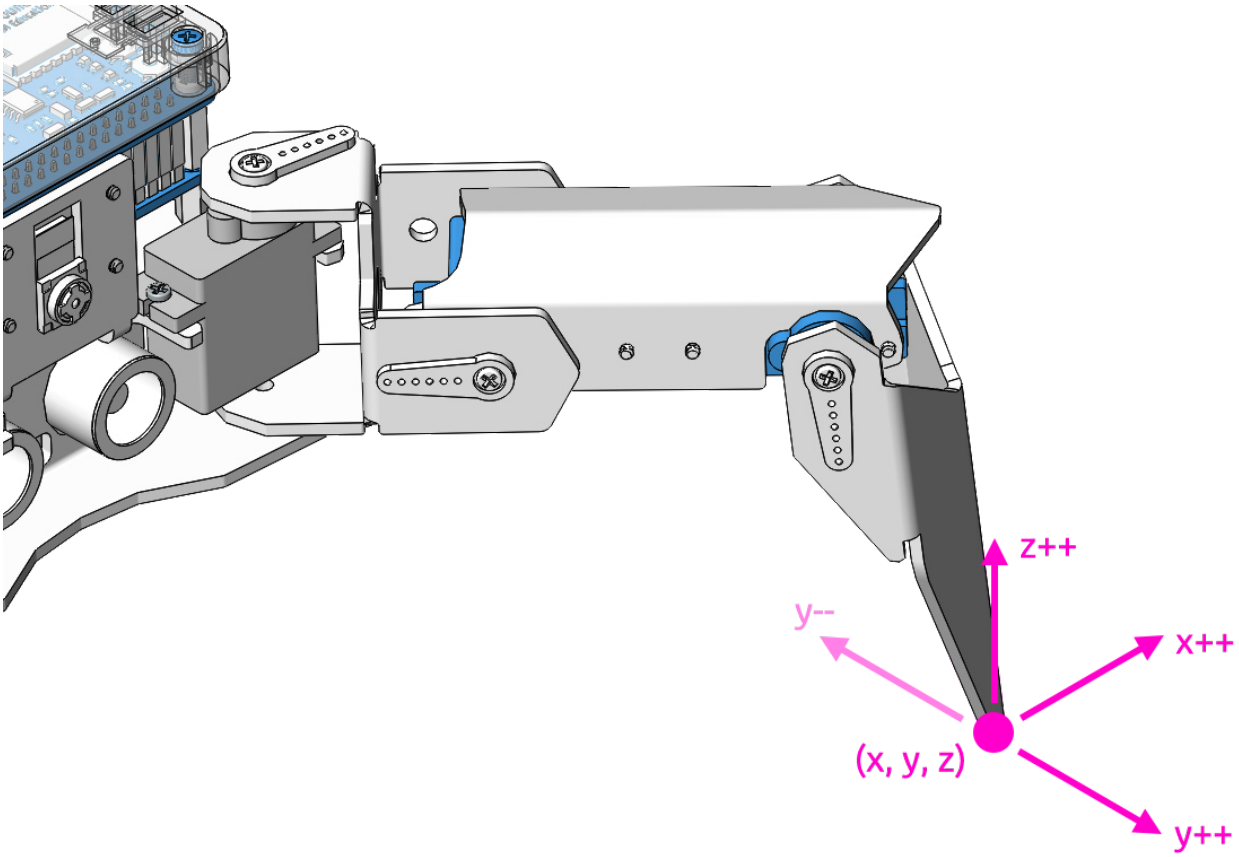
其实 `step_list` 在第一个方法中调用字符串，其本质也是通过字典来接收包含四个坐标值的数组。

```
step_list = {
    "stand":[
        [50, 50, -80],
        [50, 50, -80],
        [50, 50, -80],
        [50, 50, -80]
    ],
    "sit":[
        [50, 50, -33],
        [50, 50, -33],
        [50, 50, -33],
        [50, 50, -33]
    ],
}
```

3.12 调整姿势

在这个例子中，我们使用键盘来控制 PiCrawler 的脚来摆出我们想要的姿势。

您可以按空格键打印出当前坐标值。当您为 PiCrawler 创建独特的动作时，这些坐标值会派上用场。



运行代码

```
cd /home/pi/picrawler/examples
sudo python3 do_single_leg.py
```

代码运行后，请根据终端弹出的提示进行操作。

- 按下 1234 来分别选择脚，1：右前脚，2：左前脚，3：左后脚，4：右后脚
- 按下 w，a，s，d，r，和 f 来慢慢控制 PiCrawler 的坐标值。
- 按下 空格键来打印所有的坐标值。
- 按下 esc 来退出。

代码

```
from picrawler import Picrawler
from time import sleep
import readchar

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
speed = 80


manual = '''
-------- PiCrawler Controller ---------
    .......           .......
    <=|   2   |┌-┌┐-┐|   1   |=>
```

(下页继续)

(续上页)

```
    `````````  ├      ┤  `````````
               ├   ┤
    ....   ├   ┤ ....
    <=| 3 |└------┘| 4 |=>
    `````````   `````````
    1: Select right front leg
    2: Select left front leg
    3: Select left rear leg
    4: Select right rear leg

    W: Y++   R: Z++
    A: X--   F: Z--
    S: Y--
    D: X++   Esc: Quit
'''
legs_list = ['right front', 'left front', 'left rear', 'right rear']

def main():
    leg = 0
    speed = 80
    step = 2
    print(manual)
    crawler.do_step('stand',speed)
    sleep(0.2)
    coordinate=crawler.current_step_all_leg_value()

    def show_info():
        print("\033[H\033[J",end='') # clear terminal windows
        print(manual)
        print('%s : %s'%(leg+1, legs_list[leg]))
        print('coordinate: %s'%(coordinate))

    show_info()

    while True:
        # readkey
        key = readchar.readkey()
        key = key.lower()
        # select the leg
        if key in ('1234'):
            leg = int(key) - 1
            show_info()
        # move
        elif key in ('wsadrf'):
            if 'w' == key:
                coordinate[leg][1]=coordinate[leg][1] + step
            elif 's' == key:
                coordinate[leg][1]=coordinate[leg][1] - step
            elif 'a' == key:
                coordinate[leg][0]=coordinate[leg][0] - step
            elif 'd' == key:
                coordinate[leg][0]=coordinate[leg][0] + step
            elif 'r' == key:
                coordinate[leg][2]=coordinate[leg][2] + step
            elif 'f' == key:
                coordinate[leg][2]=coordinate[leg][2] - step
```

(下页继续)

(续上页)

```
            crawler.do_single_leg(leg,coordinate[leg],speed)
            sleep(0.1)
            # coordinate=crawler.current_step_all_leg_value()
            show_info()
        # quit
        elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
            print("\n Quit")
            break

        sleep(0.05)


if __name__ == "__main__":
    main()
```

这个怎么运作?

在这个项目中需要注意的是以下三个函数:

```
do_single_leg(leg,coordinate,speed)
```

`do_single_leg(leg,coordinate,speed)`: 单独修改某条腿的坐标值。

3.13 记录新的动作

我们用键盘控制 PiCrawler 依次摆出几个动作，并记录下这些动作。然后让 PiCrawler 重复这些动作。

运行代码

```
cd /home/pi/picrawler/examples
sudo python3 record_new_step_by_keyboard.py
```

代码运行后，请根据终端弹出的提示进行操作。

- 按下 1234 来分别选择脚，`1`：右前脚，`2`：左前脚，`3`：左后脚，`4`：右后脚
- 按下 `w`，`a`，`s`，`d`，`r`，和 `f` 来慢慢控制 PiCrawler 的坐标值。
- 按下 空格键来打印所有的坐标值并且保存这一动作。
- 按下 `p` 来让 PiCrawler 来复现记录的动作。
- 按下 `esc` 来退出。

代码

```
from picrawler import Picrawler
from time import sleep
import sys
import tty
import termios
import copy

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])
speed = 80
```

(下页继续)

(续上页)

```
def readchar():
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return ch


manual = '''
Press keys on keyboard to control PiSloth!
    w: Y++
    a: X--
    s: Y--
    d: X++
    r: Z++
    f: Z--
    1: Select right front leg
    2: Select left front leg
    3: Select left rear leg
    4: Select right rear leg
    space: Print all leg coodinate & Save this step
    p: Play all saved step
    esc: Quit
'''


new_step=[]

def save_new_step():
    new_step.append(copy.deepcopy(crawler.current_step_all_leg_value()))
    print(new_step)

def play_all_new_step():
    for step in new_step:
        crawler.do_step(step,speed)
        sleep(0.6)

def main():

    speed = 80
    print(manual)
    crawler.do_step('sit',speed)
    leg = 0
    coodinate=crawler.current_step_leg_value(leg)
    while True:
        key = readchar()
        key = key.lower()
        # print(key)
        if 'w' == key:
            coodinate[1]=coodinate[1]+2
        elif 's' == key:
            coodinate[1]=coodinate[1]-2
        elif 'a' == key:
```

(下页继续)

(续上页)

```
            coodinate[0]=coodinate[0]-2
        elif 'd' == key:
            coodinate[0]=coodinate[0]+2
        elif 'r' == key:
            coodinate[2]=coodinate[2]+2
        elif 'f' == key:
            coodinate[2]=coodinate[2]-2
        elif '1' == key:
            leg=0
            coodinate=crawler.current_step_leg_value(leg)
        elif '2' == key:
            leg=1
            coodinate=crawler.current_step_leg_value(leg)
        elif '3' == key:
            leg=2
            coodinate=crawler.current_step_leg_value(leg)
        elif '4' == key:
            leg=3
            coodinate=crawler.current_step_leg_value(leg)
        elif chr(32) == key:
            print("[[right front],[left front],[left rear],[right rear]]")
            print("saved new step")
            print(crawler.current_step_all_leg_value())
            save_new_step()
        elif 'p' == key:
            play_all_new_step()
        elif chr(27) == key:# 27 for ESC
            break

        sleep(0.05)
        crawler.do_single_leg(leg,coodinate,speed)
    print("\n q Quit")


if __name__ == "__main__":
    main()
```

这个怎么运作?

这个项目参考自调整姿势。我们增加了记录和回放功能。

记录功能由以下代码实现。

```
new_step=[]

def save_new_step():
    new_step.append(copy.deepcopy(crawler.current_step_all_leg_value()))
    print(new_step)
```

注解: 这里的赋值需要用到 Deep Copy 函数, 否则赋值的时候 new_step 将不会被分配新的数组对象。

回放功能由以下代码实现。

```
def play_all_new_step():
    for step in new_step:
```

(下页继续)

(续上页)

```
        crawler.do_step(step,speed)
        sleep(0.6)
```

3.14 组合动作

我们已经知道如何让 PiCrawler 摆出一个特定的姿势，下一步就是将这些姿势组合起来形成一个连续的动作。这个项目中，我们让 PiCrawler 的四只脚上下摆动，随着音乐跳跃。

运行代码

```
cd /home/pi/picrawler/examples
sudo python3 twist.py
```

代码运行后，PiCrawler 的四只脚上下摆动，随着音乐跳跃。

代码

```
from picrawler import Picrawler
from time import sleep
from robot_hat import Music

music = Music()
crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])


def twist(speed):
     ## [right front],[left front],[left rear],[left rear]
    new_step=[[50, 50, -80], [50, 50, -80],[50, 50, -80], [50, 50, -80]]
    for i in range(4):
        for inc in range(30,60,5):
            rise = [50,50,(-80+inc*0.5)]
            drop = [50,50,(-80-inc)]

            new_step[i]=rise
            new_step[(i+2)%4] = drop
            new_step[(i+1)%4] = rise
            new_step[(i-1)%4] = drop
            crawler.do_step(new_step,speed)

def main():

    music.background_music('./musics/sports-Ahjay_Stelino.mp3')
    music.music_set_volume(20)

    while True:
        twist(speed=100)


if __name__ == "__main__":
    main()
```

这个怎么运作?

在这段代码中，需要注意下面这个部分:

```
def twist(speed):
    ## [right front],[left front],[left rear],[right rear]
    new_step=[[50, 50, -80], [50, 50, -80],[50, 50, -80], [50, 50, -80]]
    for i in range(4):
        for inc in range(30,60,5):
            rise = [50,50,(-80+inc*0.5)]
            drop = [50,50,(-80-inc)]

            new_step[i]=rise
            new_step[(i+2)%4] = drop
            new_step[(i+1)%4] = rise
            new_step[(i-1)%4] = drop
            crawler.do_step(new_step,speed)
```

简单的说，就是使用两层 for 循环，让 new_step 数组产生连续有规律的变化，同时用 crawler.do_step() 函数执行姿势，形成连续动作。

你可以从调整姿势 课程中直观的获得每个动作对应的坐标值数组。

除此以外，该示例还可以播放背景音乐。实现方法如下。

通过导入以下库来播放音乐。

```
from robot_hat import Music
```

声明一个 Music 对象。

```
music = Music()
```

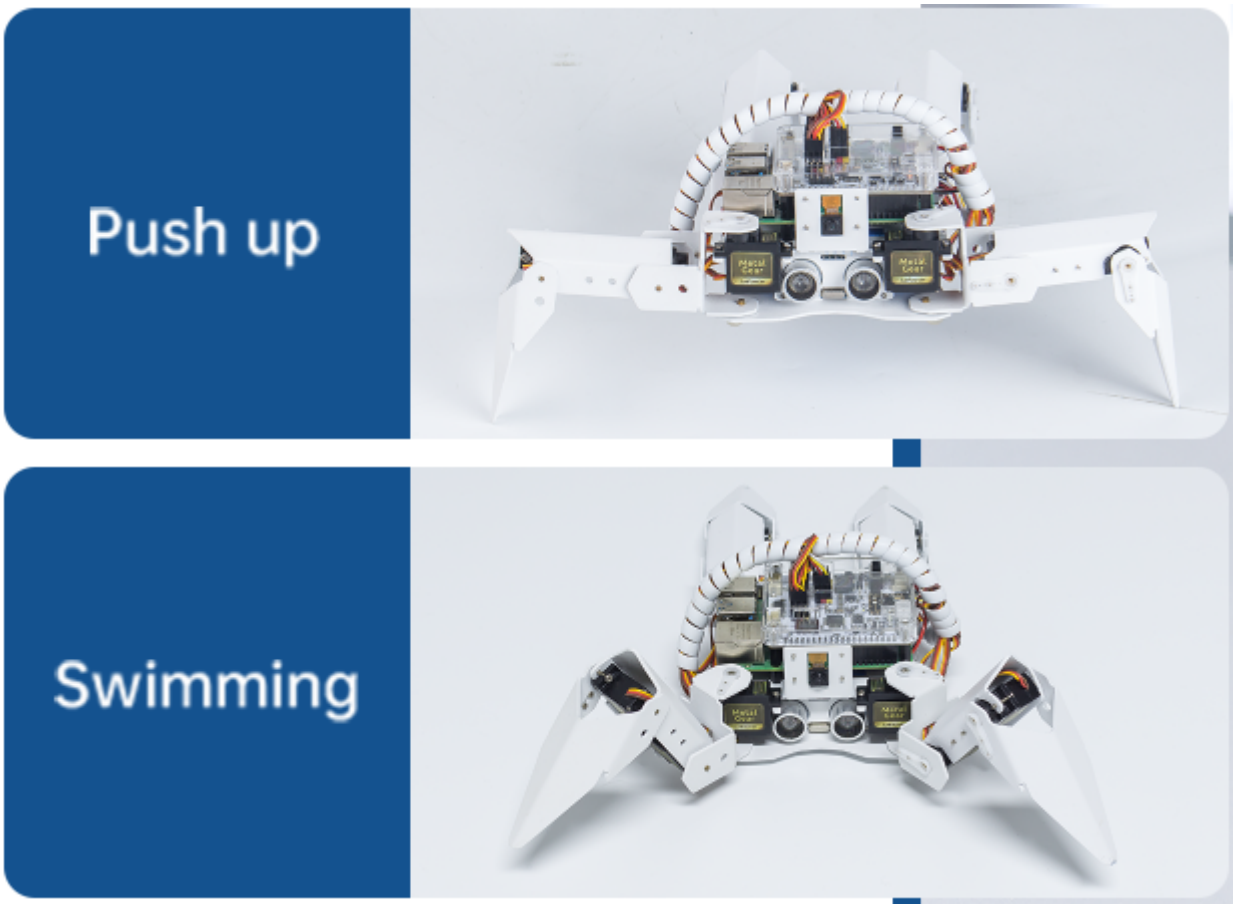
播放 picrawler/examples/musics 目录中的背景音乐，音量设置为 20。您也可以通过*Filezilla 软件* 软件将音乐添加到 musics 文件夹中。

```
music.background_music('./musics/sports-Ahjay_Stelino.mp3')
music.music_set_volume(20)
```

注解： 您可以通过*Filezilla 软件* 软件向 musics 或 sounds 文件夹添加不同的音效或音乐。

3.15 情感机器人

这个例子展示了 PiCrawler 的几个有趣的自定义动作。



运行代码

```
cd /home/pi/picrawler/examples
sudo python3 emotional_robot.py
```

代码运行后，你会看到 PiCrawler 做一些有趣的动作，比如游泳，俯卧撑，摆手和摇摆。

代码

```
from picrawler import Picrawler
from time import sleep

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

def handwork(speed):

    basic_step = []
    basic_step = crawler.step_list.get("sit")
    left_hand  = crawler.mix_step(basic_step,0,[0,50,80])
    right_hand  = crawler.mix_step(basic_step,1,[0,50,80])
    two_hand = crawler.mix_step(left_hand,1,[0,50,80])
```

(下页继续)

(续上页)

```
    crawler.do_step('sit',speed)
    sleep(0.6)
    crawler.do_step(left_hand,speed)
    sleep(0.6)
    crawler.do_step(two_hand,speed)
    sleep(0.6)
    crawler.do_step(right_hand,speed)
    sleep(0.6)
    crawler.do_step('sit',speed)
    sleep(0.6)

def twist(speed):

    new_step=[[50, 50, -80], [50, 50, -80],[50, 50, -80], [50, 50, -80]]
    for i in range(4):
        for inc in range(30,60,5):
            rise = [50,50,(-80+inc*0.5)]
            drop = [50,50,(-80-inc)]

            new_step[i]=rise
            new_step[(i+2)%4] = drop
            new_step[(i+1)%4] = rise
            new_step[(i-1)%4] = drop
            crawler.do_step(new_step,speed)

##"[[right front], [left front], [left rear], [left rear]]")

def pushup(speed):
    up=[[80, 0, -100], [80, 0, -100],[0, 120, -60], [0, 120, -60]]
    down=[[80, 0, -30], [80, 0, -30],[0, 120, -60], [0, 120, -60]]
    crawler.do_step(up,speed)
    sleep(0.6)
    crawler.do_step(down,speed)
    sleep(0.6)

def swimming(speed):
    for i in range(100):
        crawler.do_step([[100-i,i,0],[100-i,i,0],[0,120,-60+i/5],[0,100,-40-i/5]],
↪speed)

# main
def main():
    speed = 100

    swimming(speed)
    pushup(speed)
    handwork(speed)
    twist(speed)

    sleep(0.05)

if __name__ == "__main__":
    main()
```

CHAPTER 4

玩转 EzBlock

EzBlock 是 SunFounder 所使用的一个软件开发平台，用于帮助初学者和新手在树莓派平台上开始学习编程。Ezbock 有两种编程语言环境：一种是图形用户界面环境，一种是命令行 Python 环境。

EzBlock 几乎可用于所有类型的设备，包括 Mac、PC 和 Android。

在使用 EzBlock 之前，请先按照以下教程来完成 EzBlock 下载，安装和使用。

4.1 EzBlock 快速指南

这里是使用 EzBlock 的一些使用指南。在组装的时候，你只需要完成安装 *EzBlock* 镜像 和舵机调零。

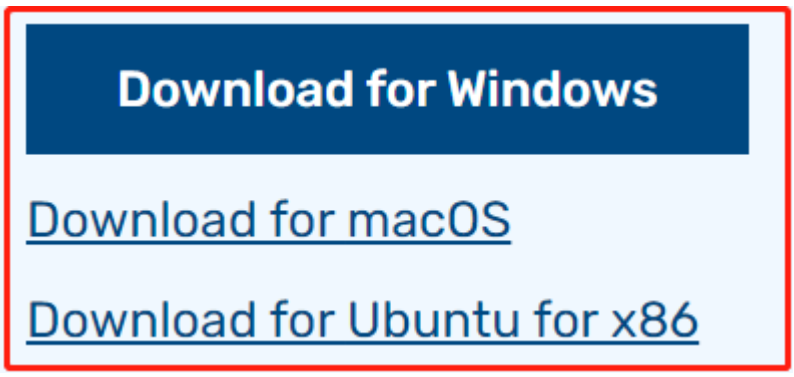
组装完成后再去下载 *EzBlock Studio*, 完成快速配置 后，就可以根据教程内容对你的机器人进行编程了。

建议按顺序阅读课程内容。

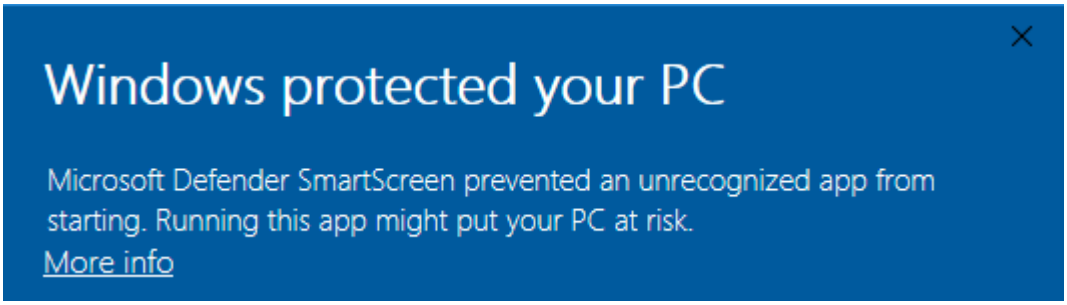
4.1.1 安装 EzBlock 镜像

在组装机器人之前，按照下面的步骤给树莓派的 Micro SD 卡烧录 EzBlock 系统。

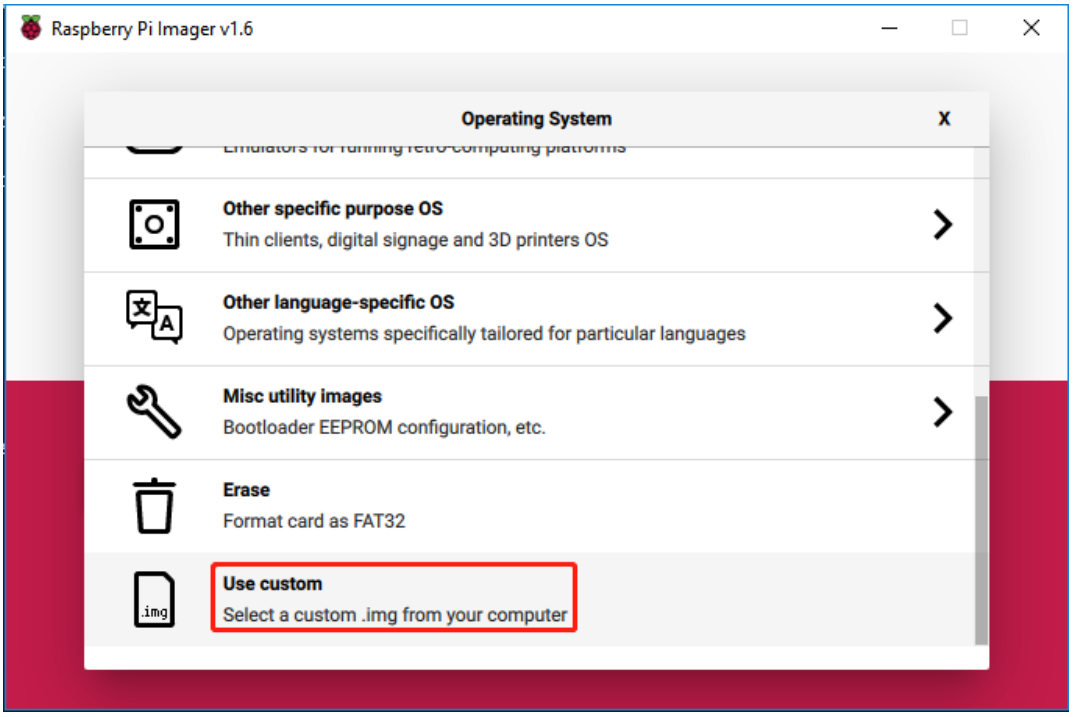
1. 树莓派开发了一个图形 SD 卡写入工具，适用于 Mac OS、Ubuntu 18.04 和 Windows，对于大多数用户来说是最简单的选择，因为它会下载映像并将其自动安装到 SD 卡。访问下载页面：https://www.raspberrypi.org/software/。单击与您的操作系统匹配的 Raspberry Pi Imager 链接，下载完成后，单击它以启动安装程序。



2. 当您启动安装程序时，您的操作系统可能会尝试阻止您运行它。例如，在 Windows 上，我收到以下消息：如果出现此消息，请点击 **更多信息**，然后点击 **仍然运行**，然后按照说明安装 Raspberry Pi Imager。

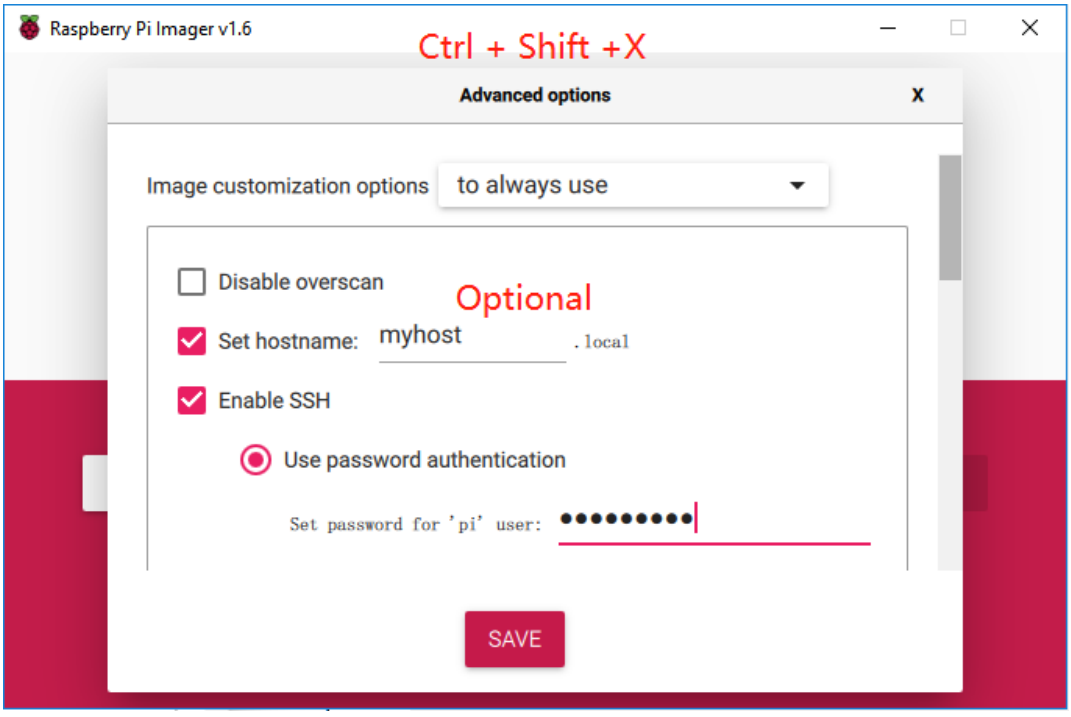


3. 将 SD 卡用读卡器插入计算机或笔记本电脑的 SD 卡插槽。
4. 下载 EzBlock 镜像
    - 天翼网盘：链接：https://cloud.189.cn/t/QNrEJnEjYNRz
    - 百度网盘：链接：https://pan.baidu.com/s/1ku1VoukCebChq9-OzkHf_g?pwd=ezbl，提取码：ezbl。
    - 由于文件超过 1G，需要在电脑上下载客户端之后才能下载文件。
5. 在 Raspberry Pi Imager 中选择刚下载的 EzBlock 镜像。



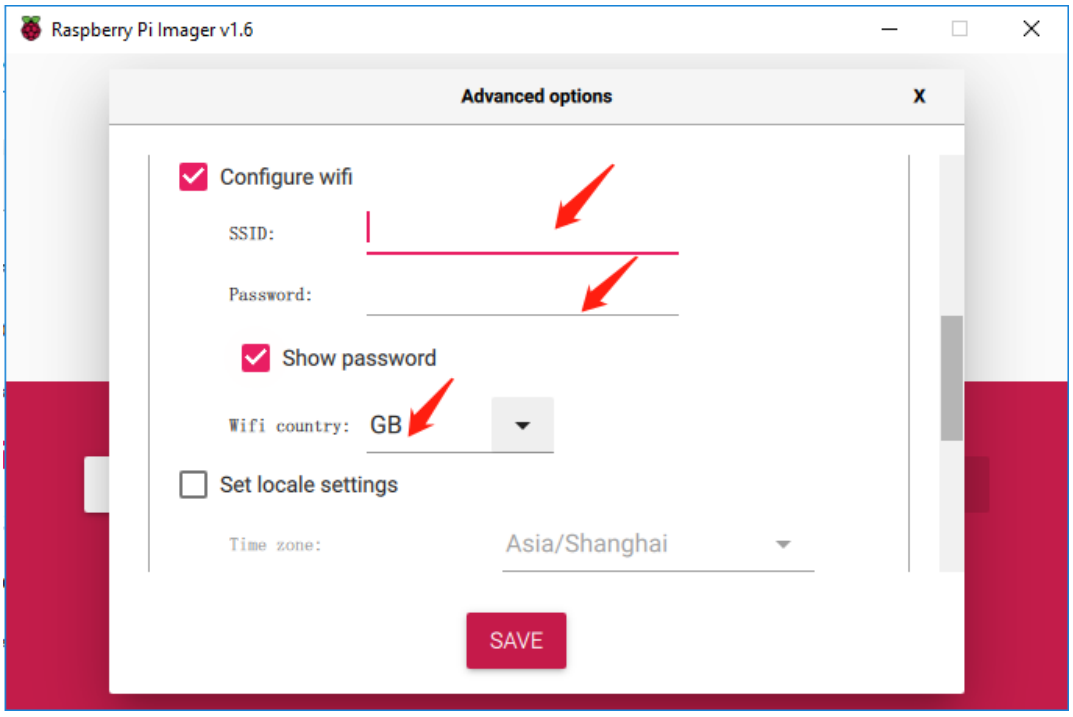
6. 按下 **Ctrl+Shift+X** 或者点击 **设置**按钮来打开 **Advanced options** 页面来设置 `hostname` 和启动 `SSH`。你可以选择“always use this image customization options”（始终使用该定制选项）。

注解： `hostname` 是让你在使用 web_ezblock, 可以用它来连接到你的产品，你也可以不设置。

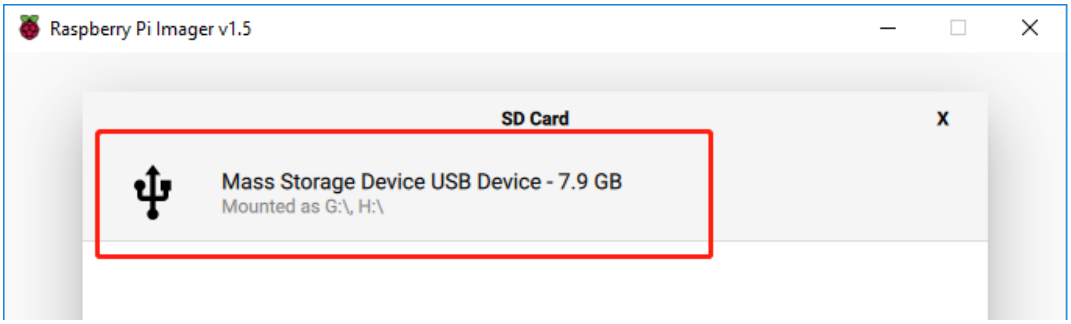


7. 下拉到底完成 WiFi 配置，然后点击 **SAVE**。

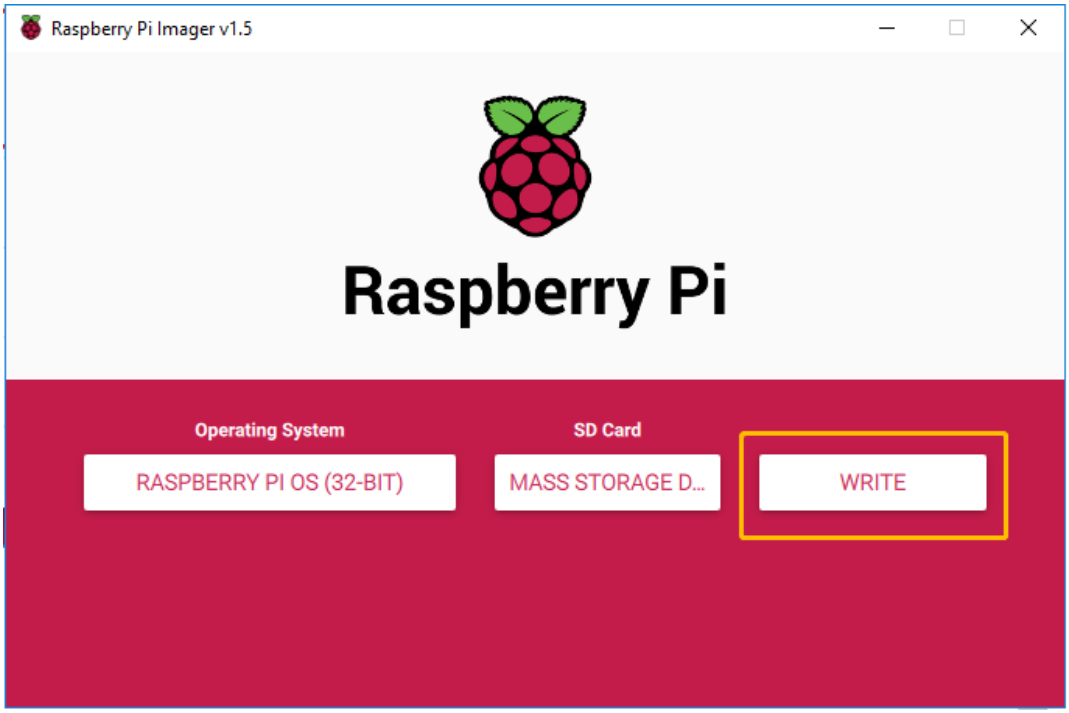
注解： **wifi country** 选择 **CN**



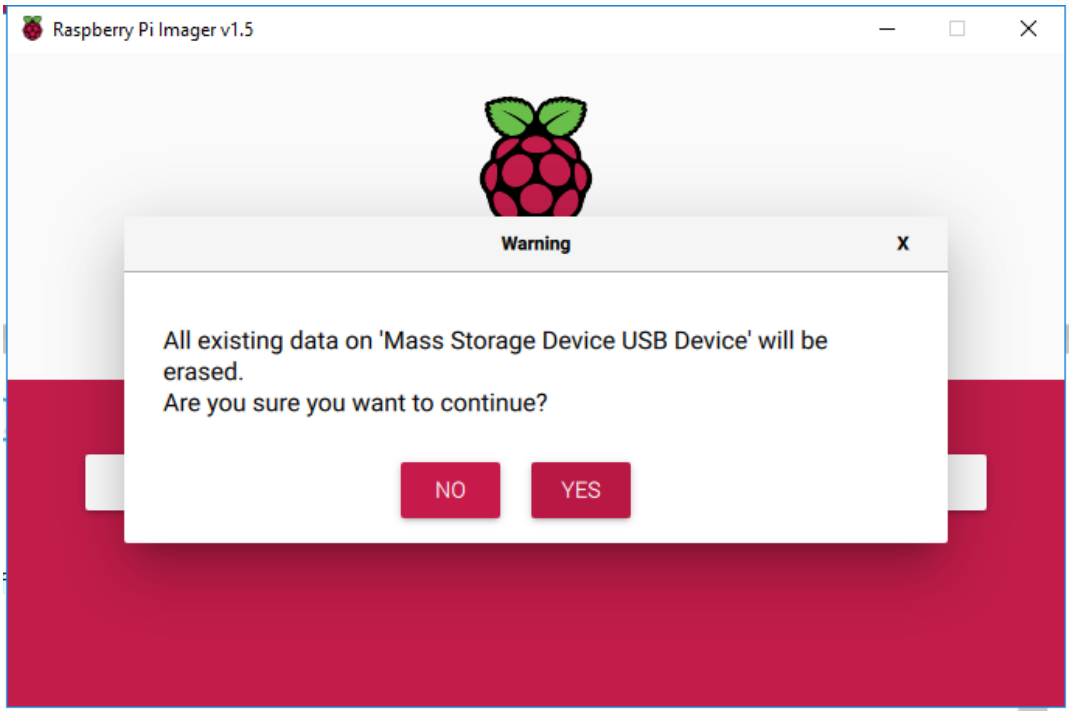
8. 选择您正在使用的 SD 卡。



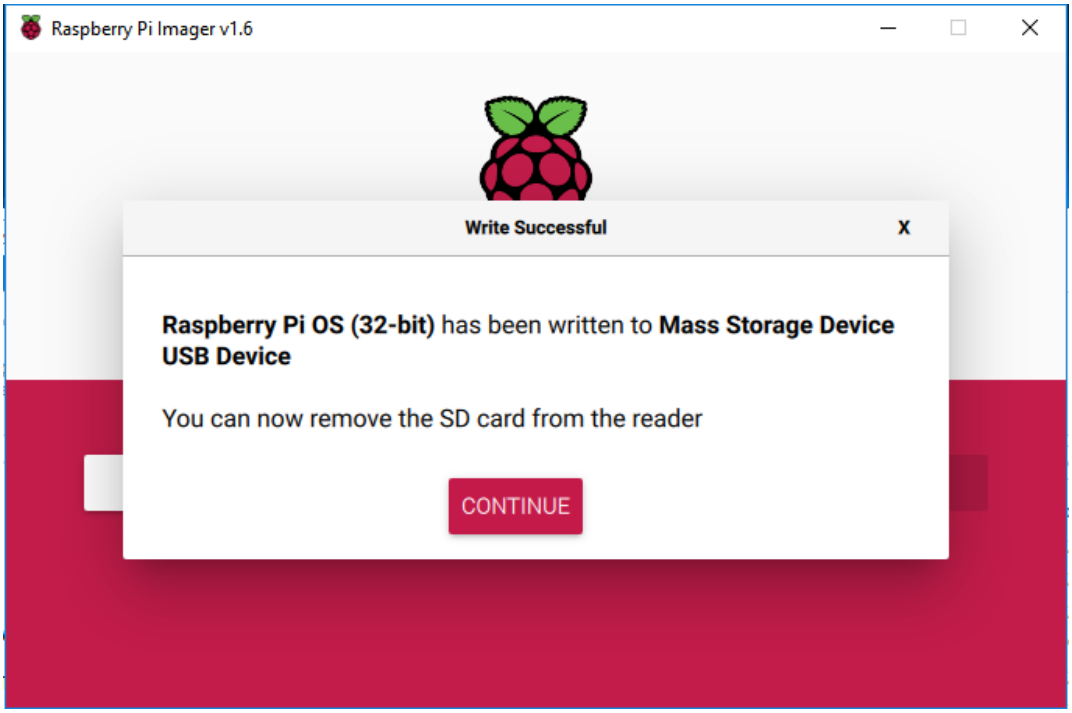
9. 单击 **WRITE** 按钮。



10. 如果您的 SD 卡上当前有任何文件，您可能希望先备份这些文件以防止永久丢失它们。如果没有要备份的文件，请单击 **YES**。



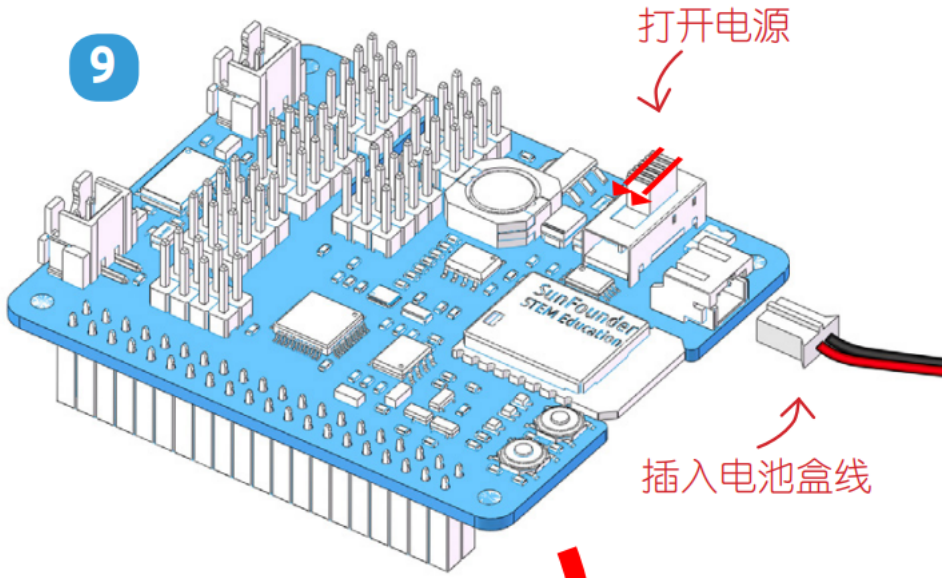
11. 等待一段时间后，会出现如下窗口，代表写入完成。现在你就可以将读卡器从电脑中拔出，并取出 SD 卡插入到树莓派上。



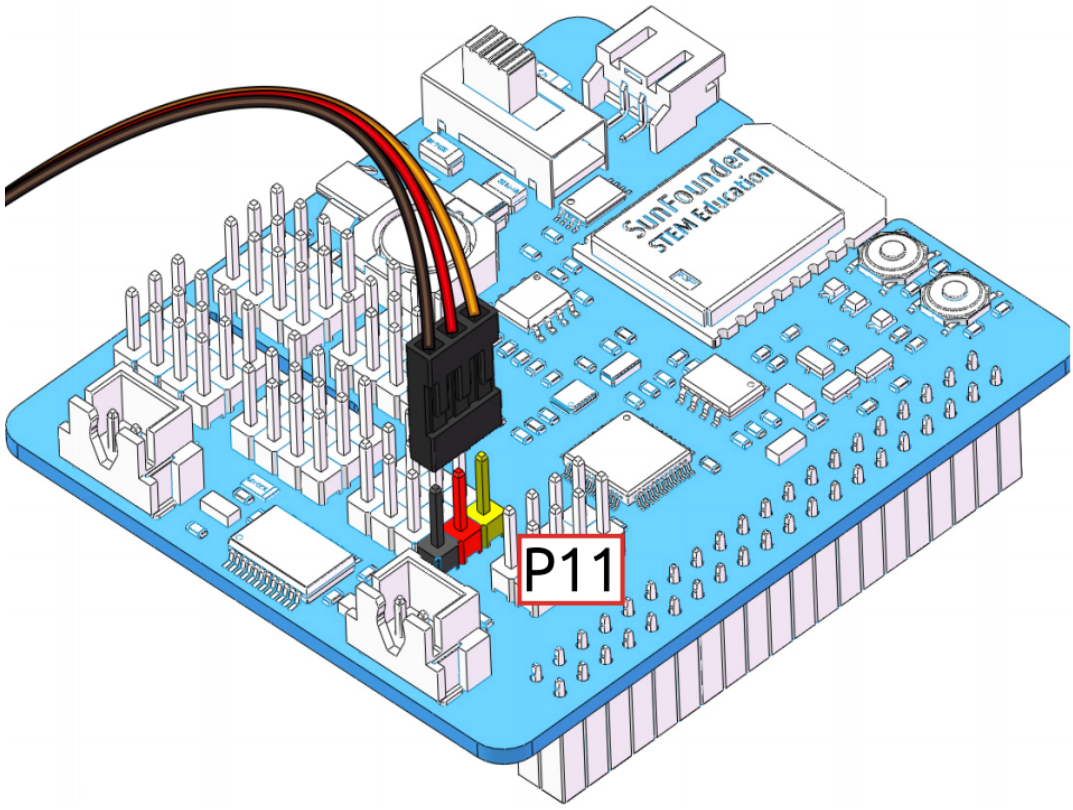
4.1.2 舵机调零

1. 为确保舵机已正确设置为 0°，首先将摇臂轻轻插入舵机轴，然后将摇臂轻轻旋转到不同的角度。

2. 按照组装折页上的提示，插入电池盒线，将电源开关拨向 ON 的位置。等待 1-2 分钟，会有声音提示树莓派启动成功。



3. 接下来，将舵机线插入 P11 端口，如下所示：



4. 此时你会看到舵机臂转动到特定的位置（0°）。如果伺服臂没有返回到 0°，请按 **RST** 按钮重新启动 Robot HAT。
5. 现在你就可以按照组装折页上的指示继续安装。

注解:

- 在用舵机螺丝固定该舵机前，不要拔出该舵机线，可在固定完之后拔出。
- 不要在上电情况下随意转动舵机以免损坏；如果舵机轴插入的角度不对，需把舵机拔出再重新插入。
- 在组装每个舵机之前，都需要将舵机电缆插入 P11 并打开电源以将它的角度设置为 0°。
- 若后面用 EzBlock APP 给机器人下载程序后，此调零功能将失效。

4.1.3 下载 EzBlock Studio

扫描下方二维码，下载 EzBlock Studio APP。

注解:

- 对于安卓手机，会提示用浏览器打开。下载成功后，根据提示允许浏览器安装该应用
- 对于 iPhone 手机，会打开 APP Store 来安装该应用。
- 对于笔记本和电脑可以使用 web_ezblock。

4.1.4 快速配置

在你第一次用 EzBlock 对你的机器人进行编程时，会有一个快速配置的过程。

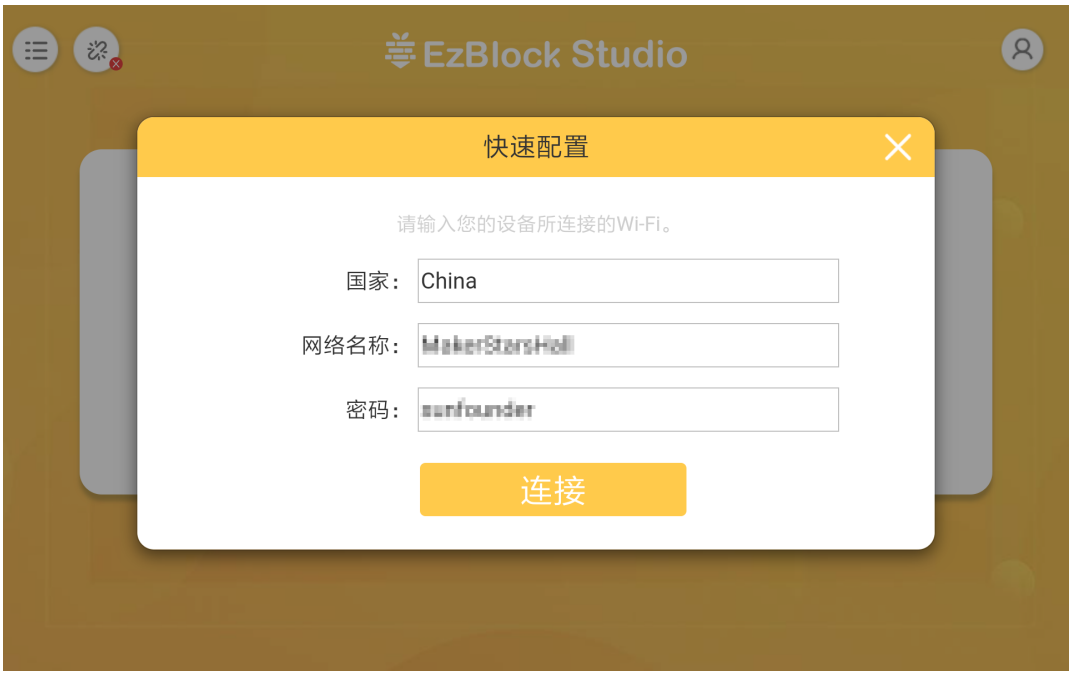
1. 打开 EzBlock Studio，会弹出一个空设备列表的窗口。您需要在打开产品电源的同时打开移动设备的蓝牙，才会出现产品编号。


```
ezb-RaspberryPi(2A:05:12:00:00:CB)
```

2. 点击右上角的 **完成**，稍等片刻，会提示 **连接成功**。此时您需要点击 **确定**来快速配置您的产品。

3. 输入您的 Wi-Fi 帐户和密码。

注解： 如果您已经提前配置了 Wi-Fi，那么 **快速配置**这一步就不会出现，直接进入下一步 **设置名称**。

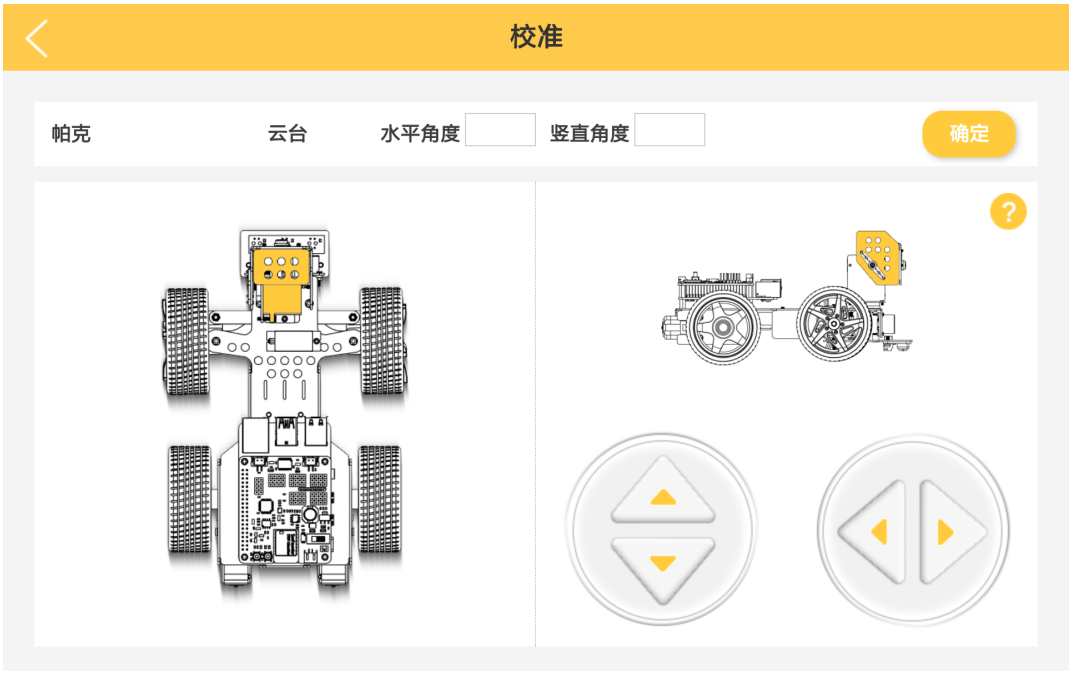


4. 选择与您匹配的产品。

5. 为您的产品输入名称。

6. 如果您的产品需要校准，会有提示告诉您可以通过点击 **立即校准**进入校准页面。如果不需要，则弹出窗口消失并返回主页。

7. 每个产品的校准页面都不一样，但是都有提示需要校准的部分。您可以点击相应的部分，然后参考 **校准帮助**进行校准。校准完成后，点击 **确认**。



4.1.5 打开或新建项目

配置完成之后，你就可以开始给你的机器人编写程序，我们已经将所有示例都已上传到 **示例**页面，你可以直接运行或打开相应的项目。或者根据课程中的提示，新建项目后，通过简单的拖拽来完成相应项目的编程。

打开项目

1. 在首页，点击 **示例**，进入示例页面。如果您只需要简单地测试这些示例，您只需点击 **运行**即可让您的产品工作。如果要查看和修改里面的代码，那么就需要点击 **编辑**进入编程页面。

2. 进入编程页面之后，将代码修改之后，点击 **下载/运行**按钮，就能看到你的机器人做出相应的效果。可点击 **运行/暂停**按钮来停止运行。

新建项目

1. 点击 **新建项目**按钮。

2. 填写项目名称。

3. 根据每个项目的提示，编写程序。

4. 点击 **下载/运行**按钮，就能看到你的机器人做出相应的效果。可点击 **运行/暂停**按钮来停止运行。

EzBlock 项目

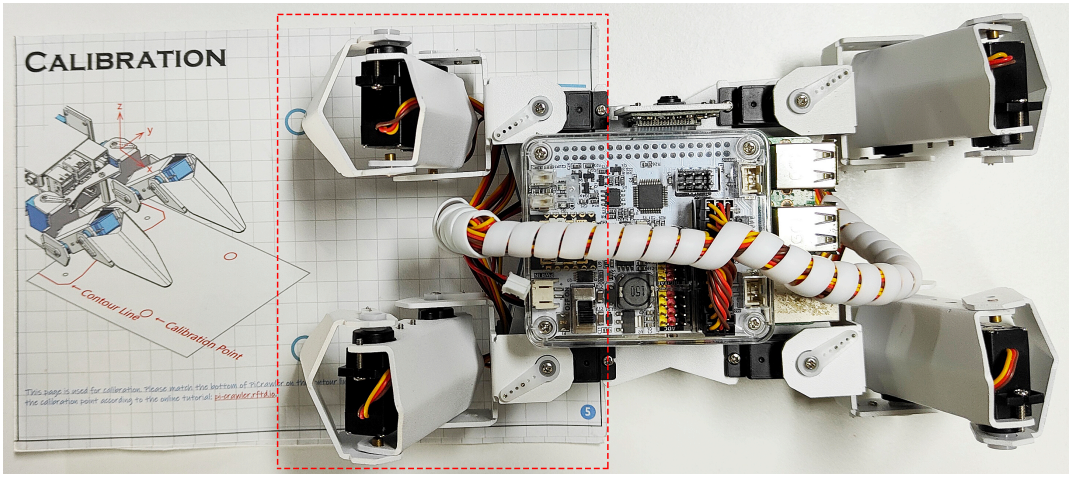
在这里，我们向您展示在 EzBlock Studio 上玩 PiCrawler 的项目。如果你是新手，可以参考各个项目里面的代码图片进行编程，可以根据提示来学习积木块的使用。

如果你不想一个一个的写这些项目，我们已经把它们上传到 EzBlock Studio 的示例页面，你可以直接运行它们，也可以在运行它们之前稍作编辑。

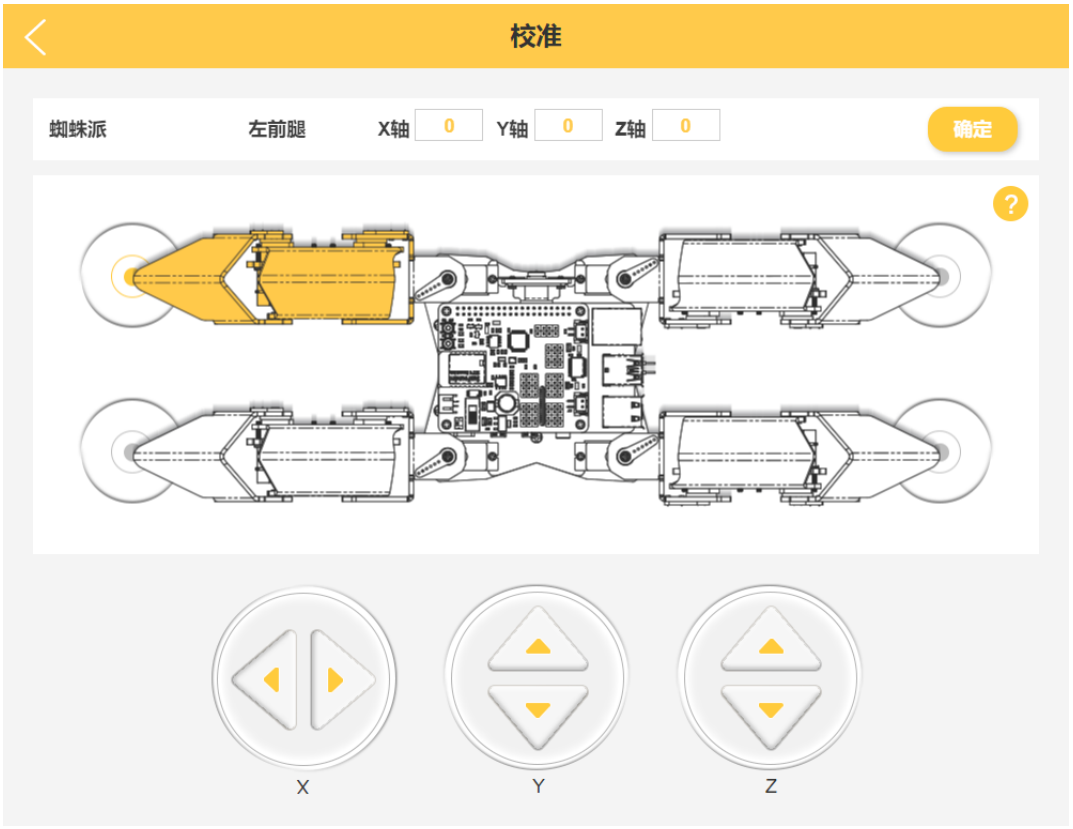
4.2 如何校准 PiCrawler

每个产品的校准页面都不一样，但是都有提示需要校准的部分。对于 PiCrawler，它的校准比较复杂。校准步骤如下：

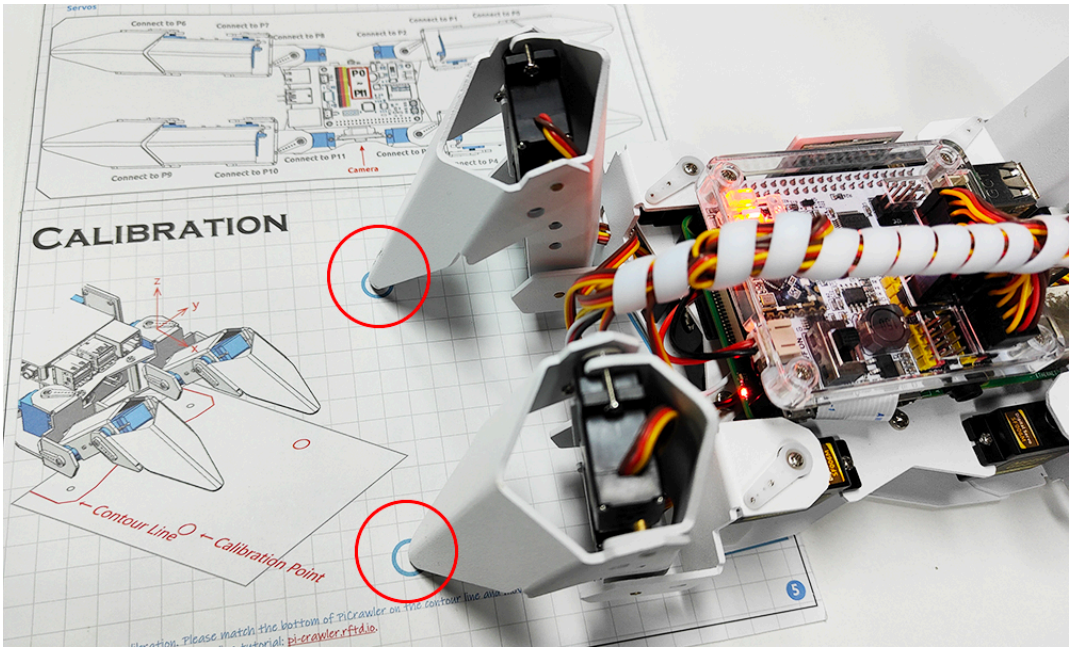
1. 拿出组装折页，将它翻到最后一页，平整的放桌上。然后将 PiCrawler 如下图放置，需要将它的底部与校准图上的轮廓线对齐。



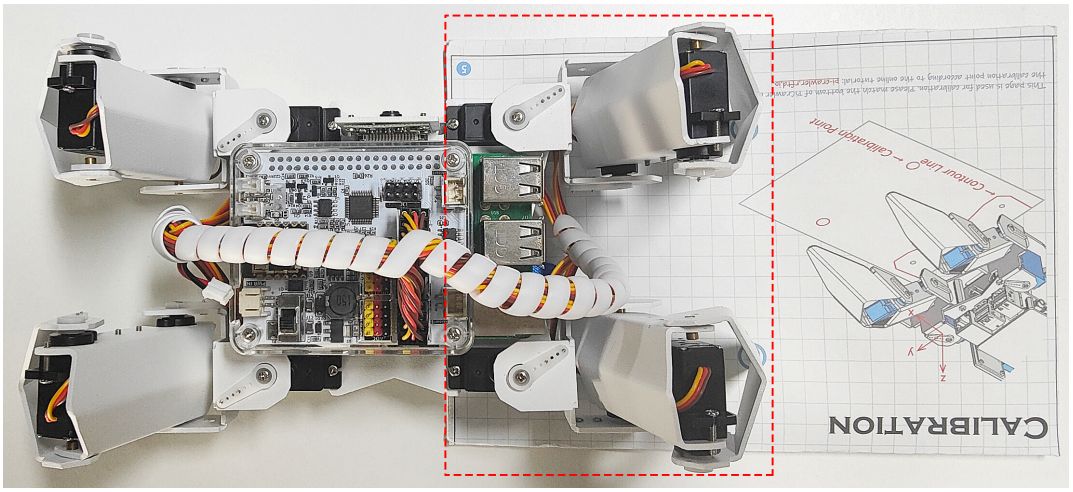
2. 回到 EzBlock，选中左侧的一只脚，然后点击 X，Y 和 Z 的 3 组按键，让脚尖慢慢对准校准点。
   - 校准按键起的是微调作用，你需要多次点击这些按键才能看到脚位变化。
   - 建议先点击 Z 轴的上键来将脚抬起来，再去调整 X 和 Y。



3. 用同样的方法校准左侧另外一只脚。



4. 校准完左侧两只脚后，将校准纸换到右边，按照上述的方法来校准右侧两只脚。

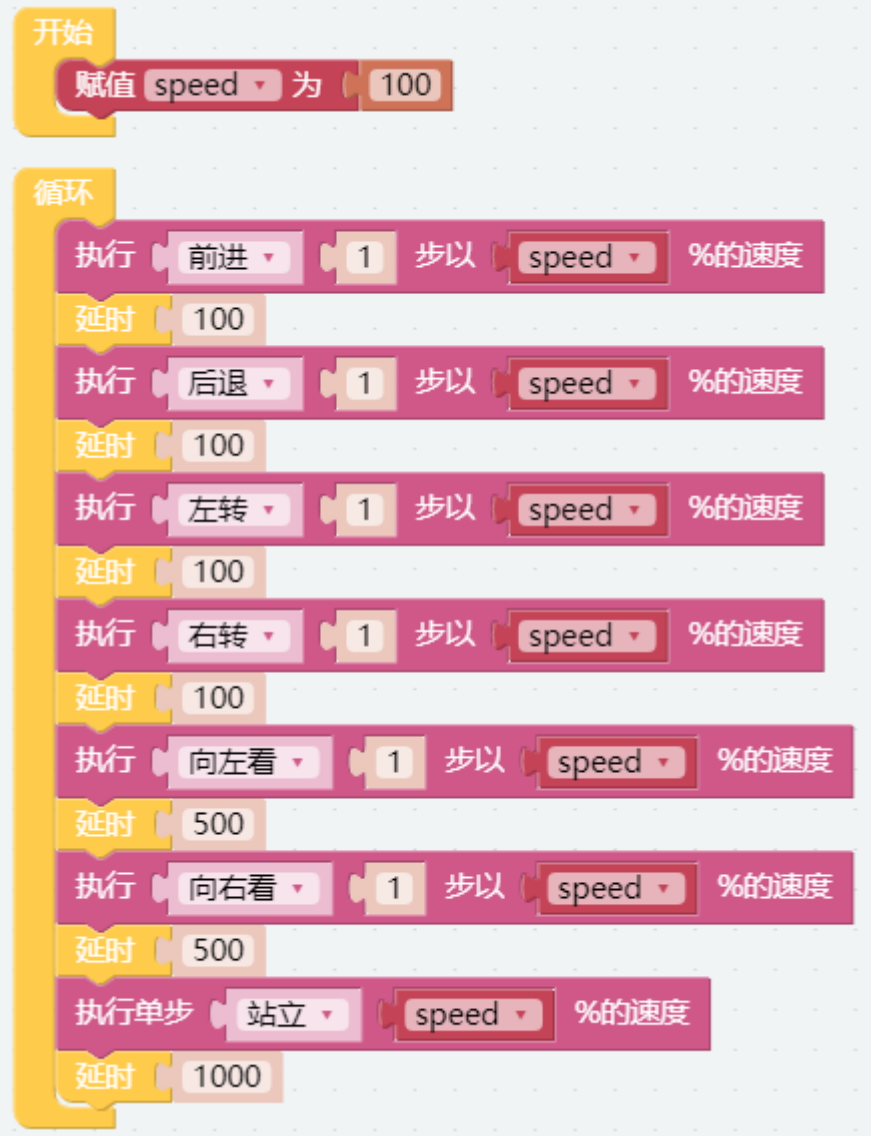


4.3 移动

这是 PiCrawler’s 的第一个项目，让它移动起来。

程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。



点击屏幕右下方的上传和运行按钮，PiCrawler 将依次执行“前进”和“后退”动作。

这个怎么运作?

首先需要了解 EzBlock 的程序框架。如下图所示:

所有 EzBlock 项目都包含这两个积木块。**开始**积木块在程序开始的时候运行，且只会运行一次，它通常用于变量的初始化；**循环**在 **开始**积木块之后运行，并且将会循环执行，通常用于实现主要的功能。删除这些积木块，只需要将它们拖拽到左侧的 **基本**类别框中。

然后来了解下面的积木块。

执行... 积木块可以让 PiCrawler 执行基本的动作。您可以修改第一个凹槽中的选项。例如，选择 “左转”、“后退” 等。第二个凹槽可以设置动作的执行次数，只能写入大于 0 的整数。第三个凹槽可以设置动作的速度，只能写 0~100 之间的整数。

执行单步... 积木块和 **执行...** 积木块类似，但它不是动作而是静态姿势。比如” 站立”，” 坐下”。

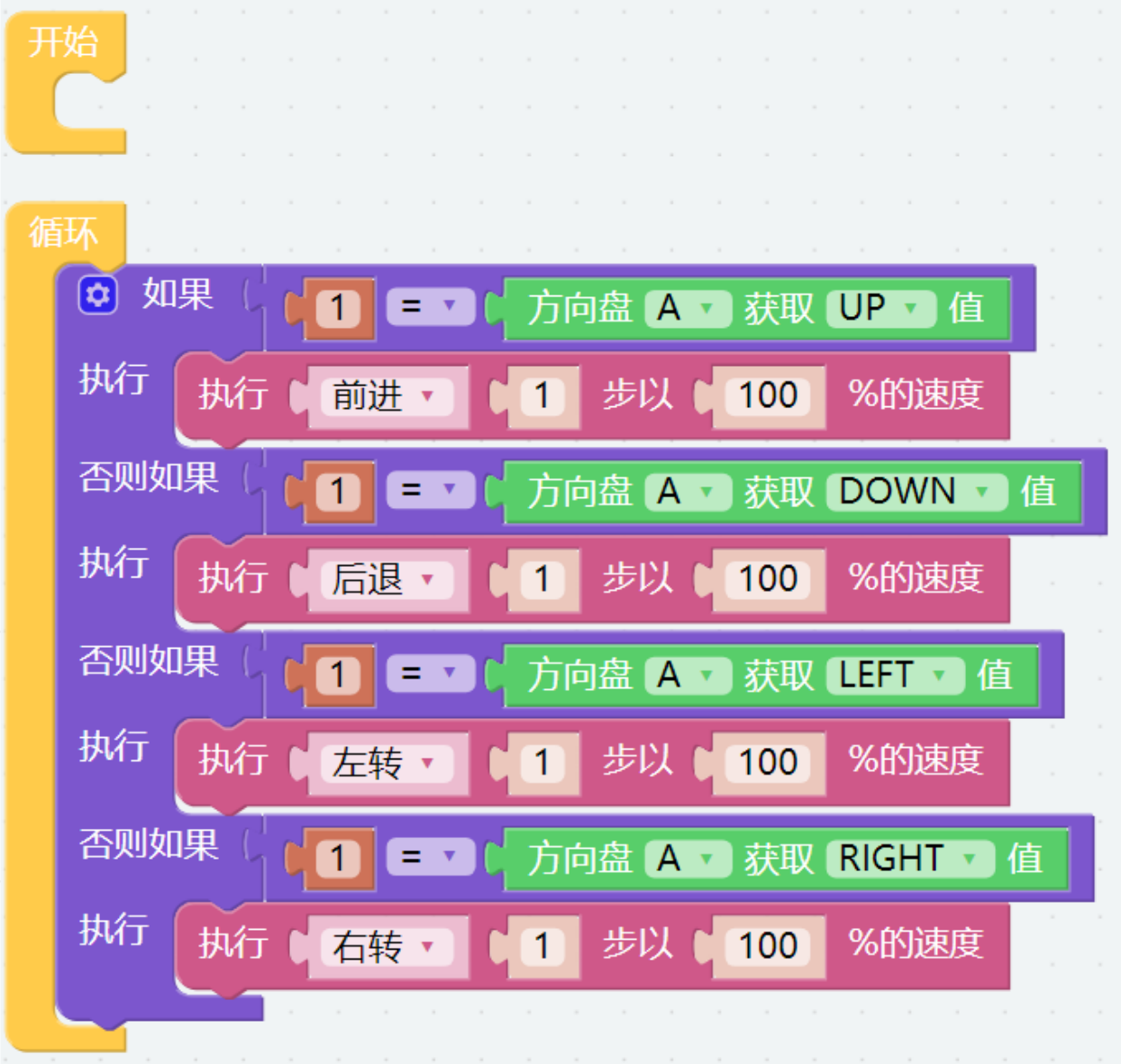
这两个块都可以从左侧的 **PiCrawler** 类别中拖出来。

4.4 远程控制

在这个项目中，我们将学习如何远程控制 PiCrawler。您可以控制 PiCrawler 向前、向后、向左和向右移动。

程序

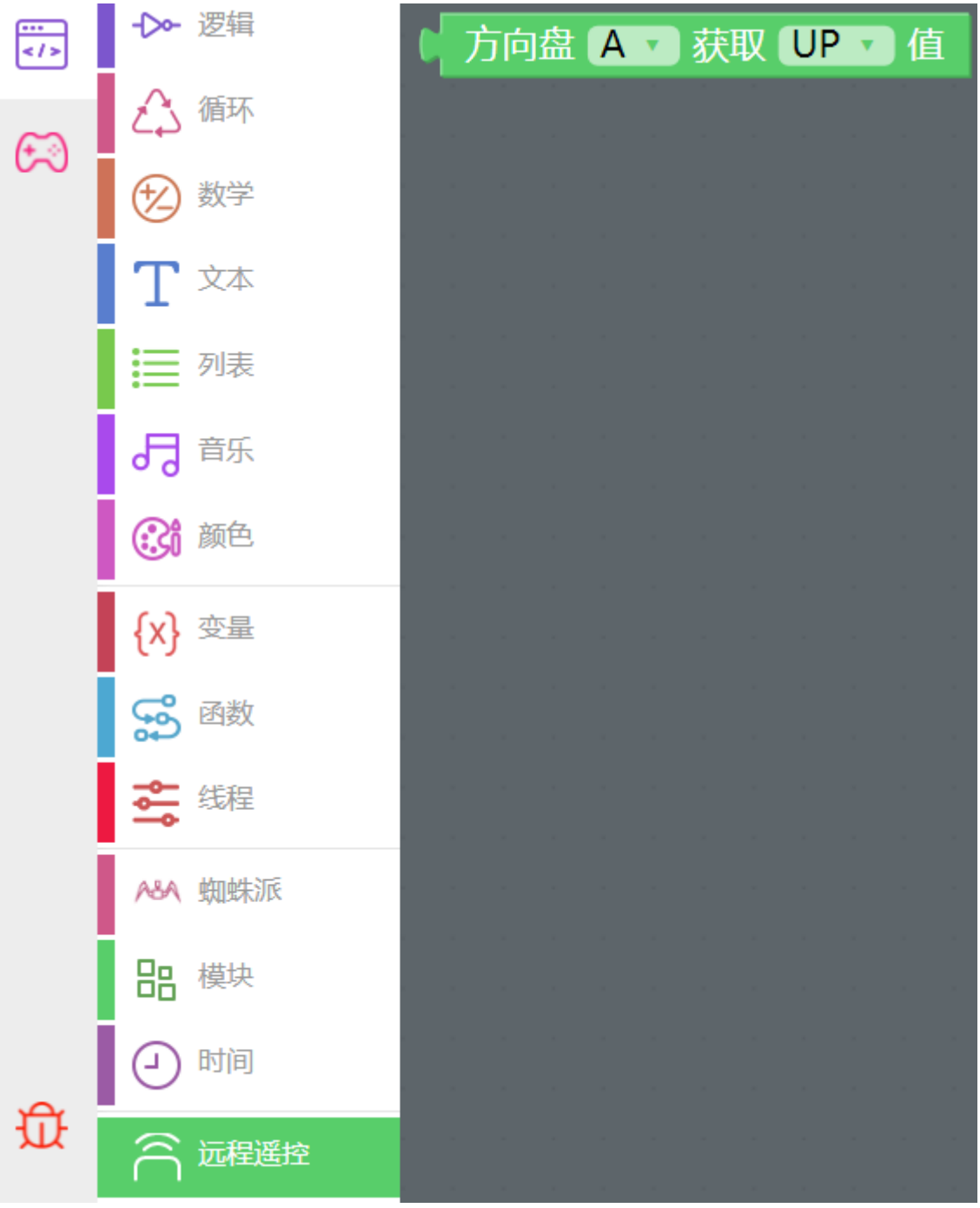
注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。



程序运行后，切换到远程控制界面，您将看到下面这个小部件。您可以通过按下方向键让 PiCrawler 动起来。

这个如何运作?

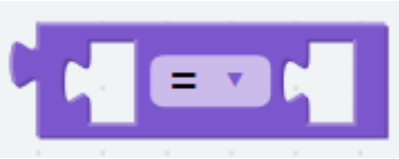
在远程控制界面上拖出小部件后，在编程界面的块类别栏中会出现一个名为 **远程遥控**的类别。
因为我们添加了小部件方向盘 A, 所以这里出现了 **方向盘 A 获取... 值**的积木块。



方向盘可以看作是一个四合一的按钮。您可以在块的第二个凹槽中选择读取哪个方向的按钮。
按下按钮时，值为“1”；没有按下按钮时，值为“0”。

我们用到了 **如果... 执行...** 的积木块 (您可以在左侧的 **逻辑**类别中找到他) 当按下方向盘的向上按钮时让 PiCrawler 向前移动一次。

你可以点击块左上角的齿轮图标修改 **如果... 执行...** 积木块的形状，实现多个判断分支。



如果... 执行... 通常和 = 一起使用, 可以通过下拉菜单将 = 条件修改为 > 或 <，具体根据实际代码修改。

4.5 音效

在本例中，我们使用 PiCrawler（准确地说是 Robot HAT）的音效。它由三部分组成，分别是背景音乐，音效和语音。

程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。

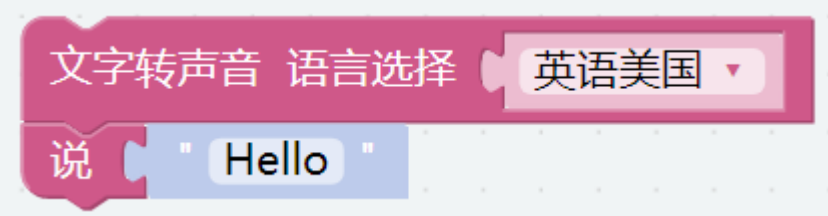
程序运行后，切换到远程控制界面，您将看到以下小部件。您可以按不同的按钮使 PiCrawler 发出不同的声音。

这个如何运作?

与背景音乐相关的功能块包括:

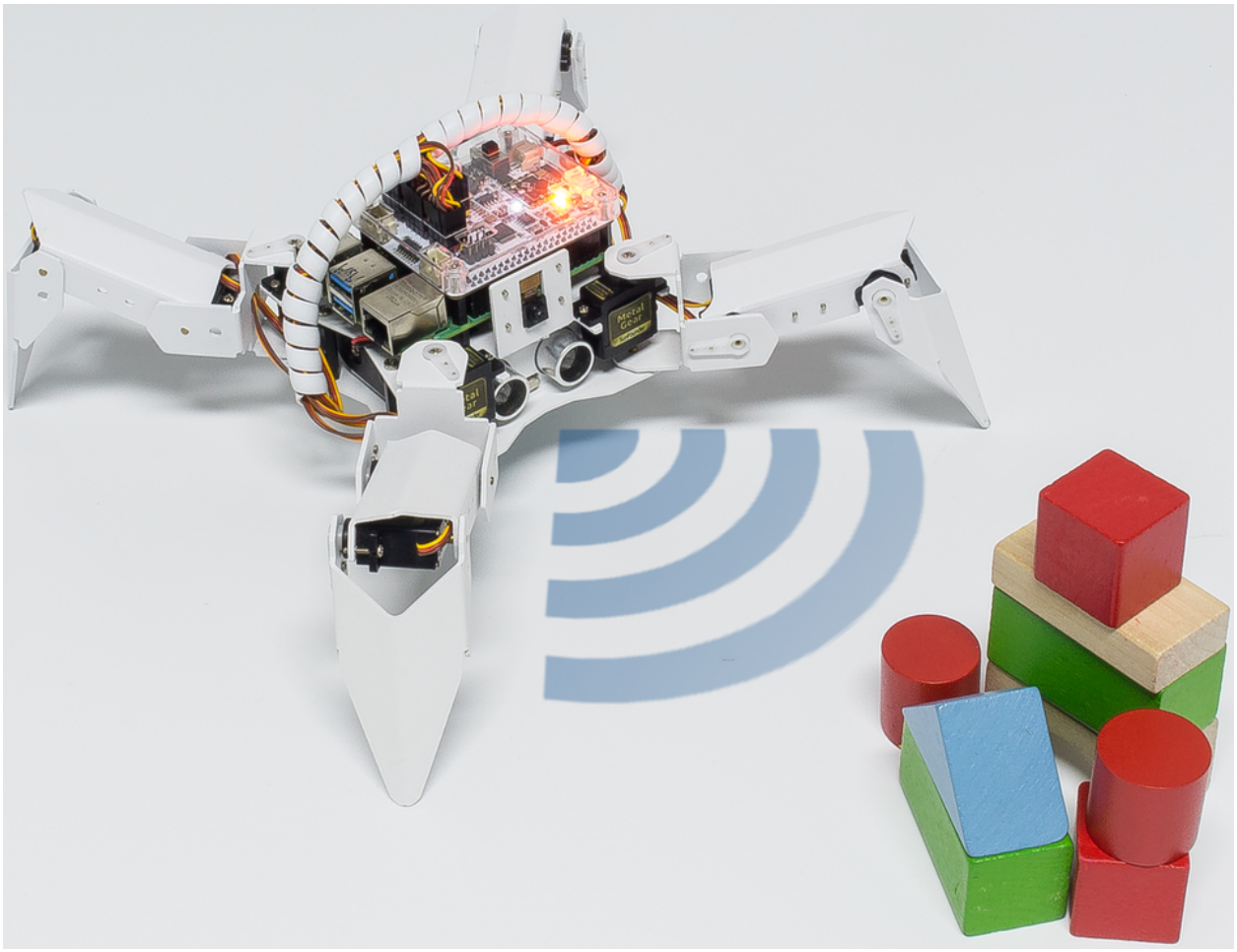
与音效相关的功能块包括:

与文本转语音相关的功能块包括:



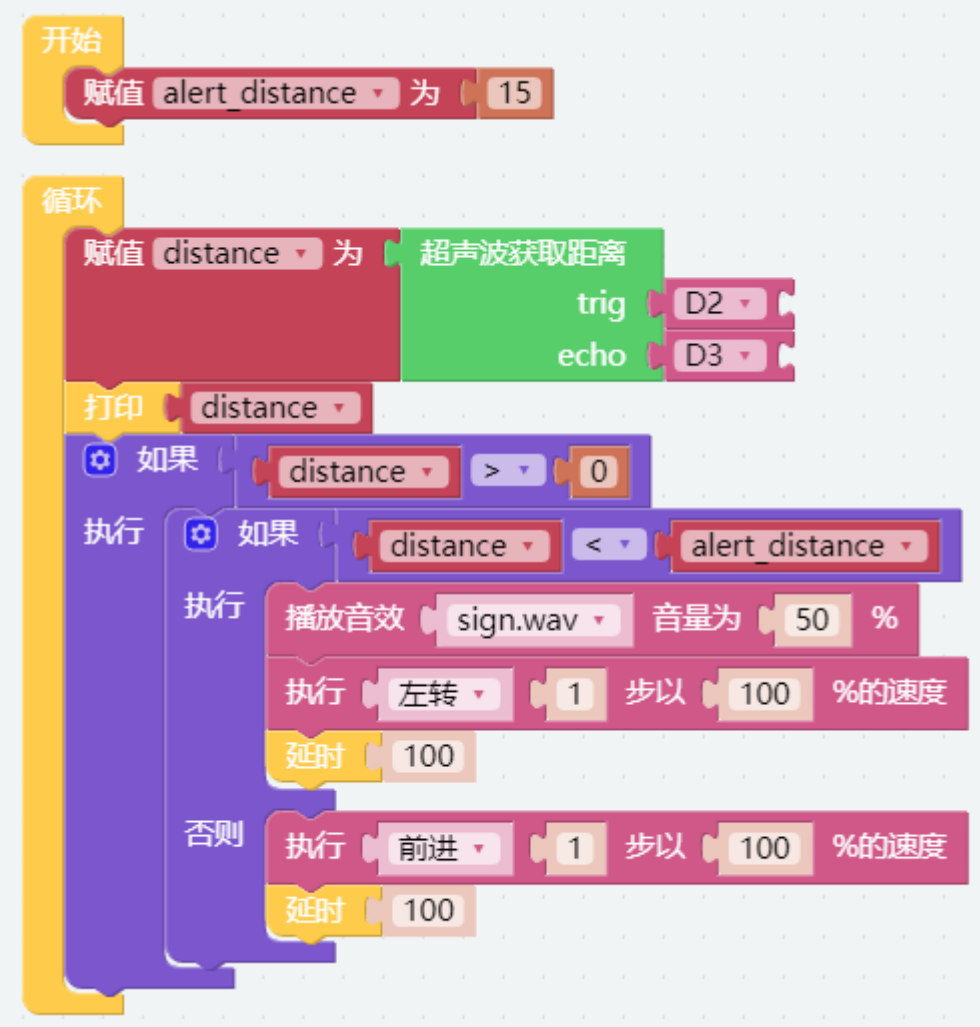
4.6 避障

在这个项目中，picrawler 将使用超声波模块来检测前方的障碍物。当 PiCrawler 检测到障碍物时，它会发出信号并寻找另一个方向前进。



程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。



程序运行后，picrawler 将使用超声波模块来检测前方的障碍物。当 PiCrawler 检测到障碍物时，它会发出信号并寻找另一个方向前进。

这个如何运作?

您可以在 **模块**类别中找到以下块来实现距离检测：

需要注意的是，此积木块的两个引脚要对应实际接线，即 trig-D2，echo-D3。

主程序逻辑如下：

- 读取超声波模块检测到的 `distance` 值，会过滤掉小于 0 的值（当超声波模块距离障碍物太远或无法正确读取数据时，会出现 `distance<0` 的情况）。

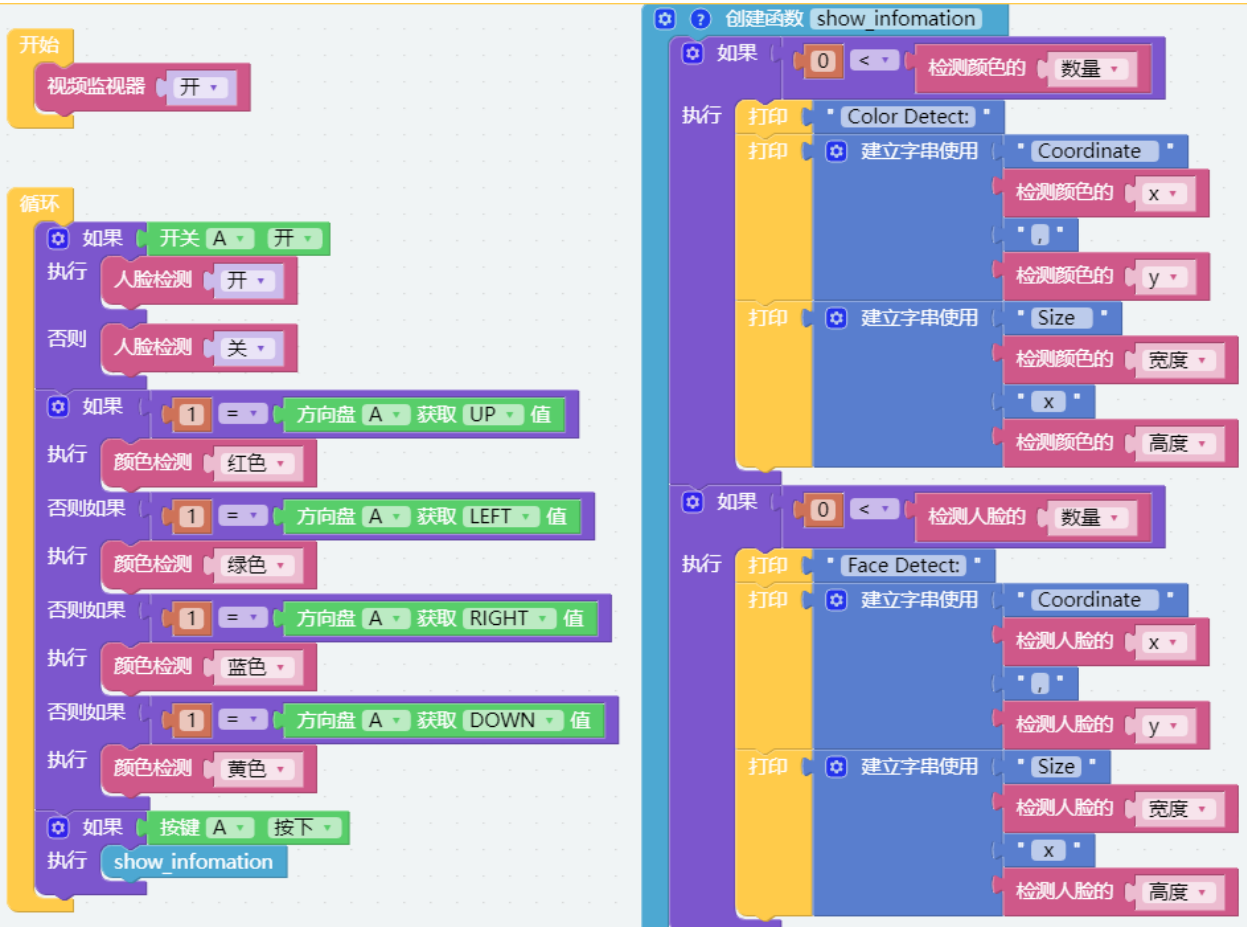
- 当 `distance` 小于 `alert_distance`（之前设置的阈值，数值为 10）时，播放音效 `sign.wav` 并让 PiCrawler 执行左转的动作。
- 当 `distance` 大于 `alert_distance` 时，让 PiCrawler 执行前进的动作。

4.7 计算机视觉

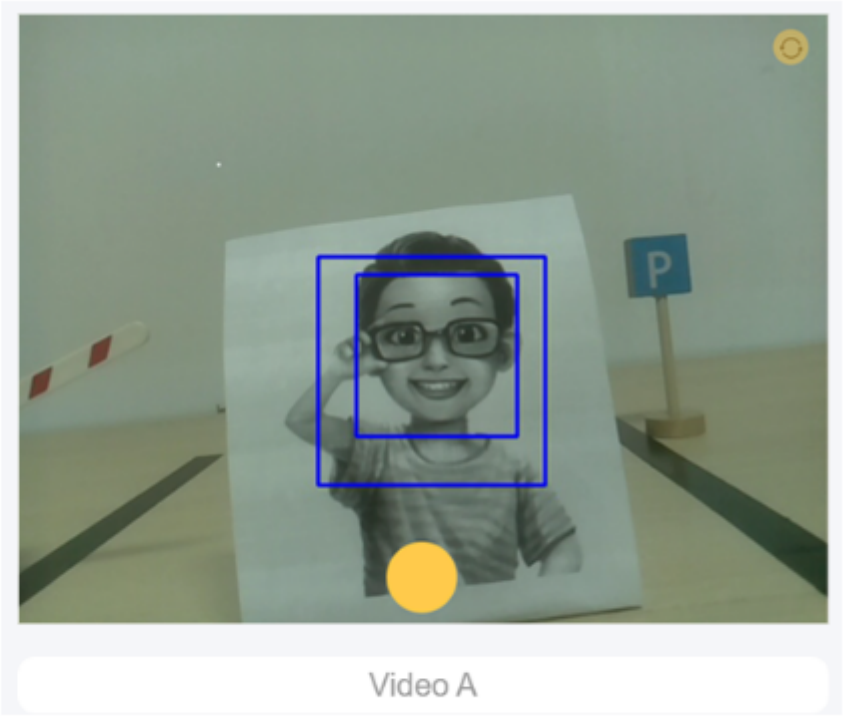
本项目将正式进入计算机视觉领域！

程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。

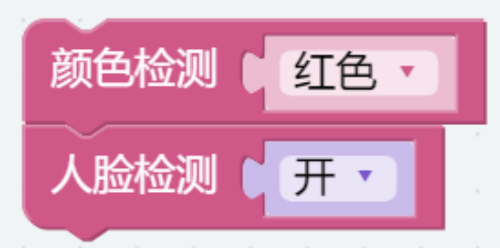


程序运行后，切换到远程控制界面，您将看到以下小部件。您可以切换开关 A 来开启/关闭人脸检测；点击方向盘 A 选择检测颜色；点击按钮 A 打印检测结果。

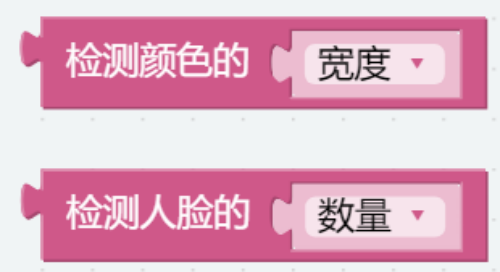


这个如何运作?

这个积木块用于启用相机模块。



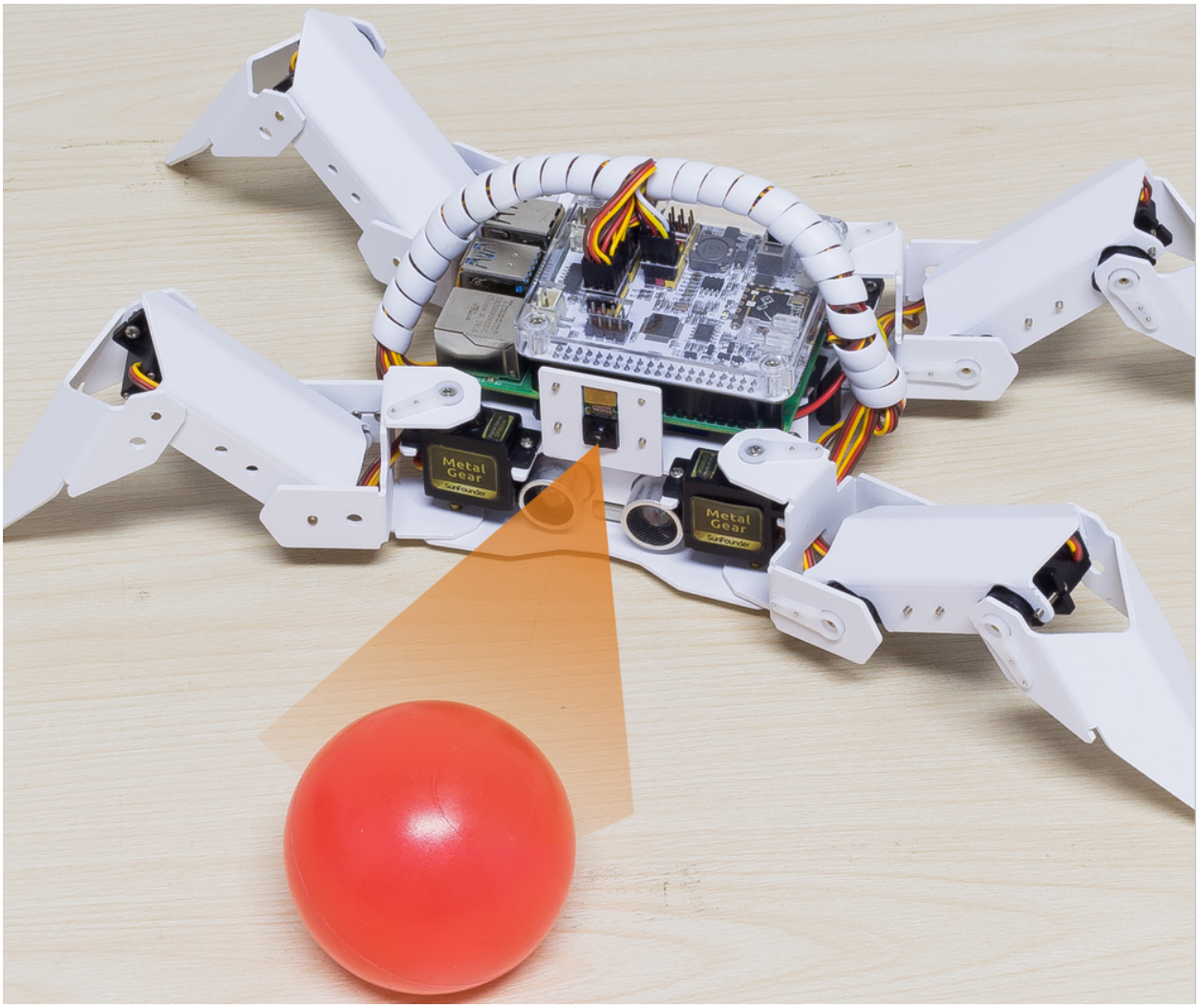
这两个块用于启用人脸检测/颜色检测功能。



这两个块用于输出信息。检测结果会输出 5 个值，分别是坐标 x 值、坐标 y 值、宽度、高度和数量。

4.8 斗牛

让 PiCrawler 成为愤怒的公牛！用它的相机追踪功能来冲撞红布！

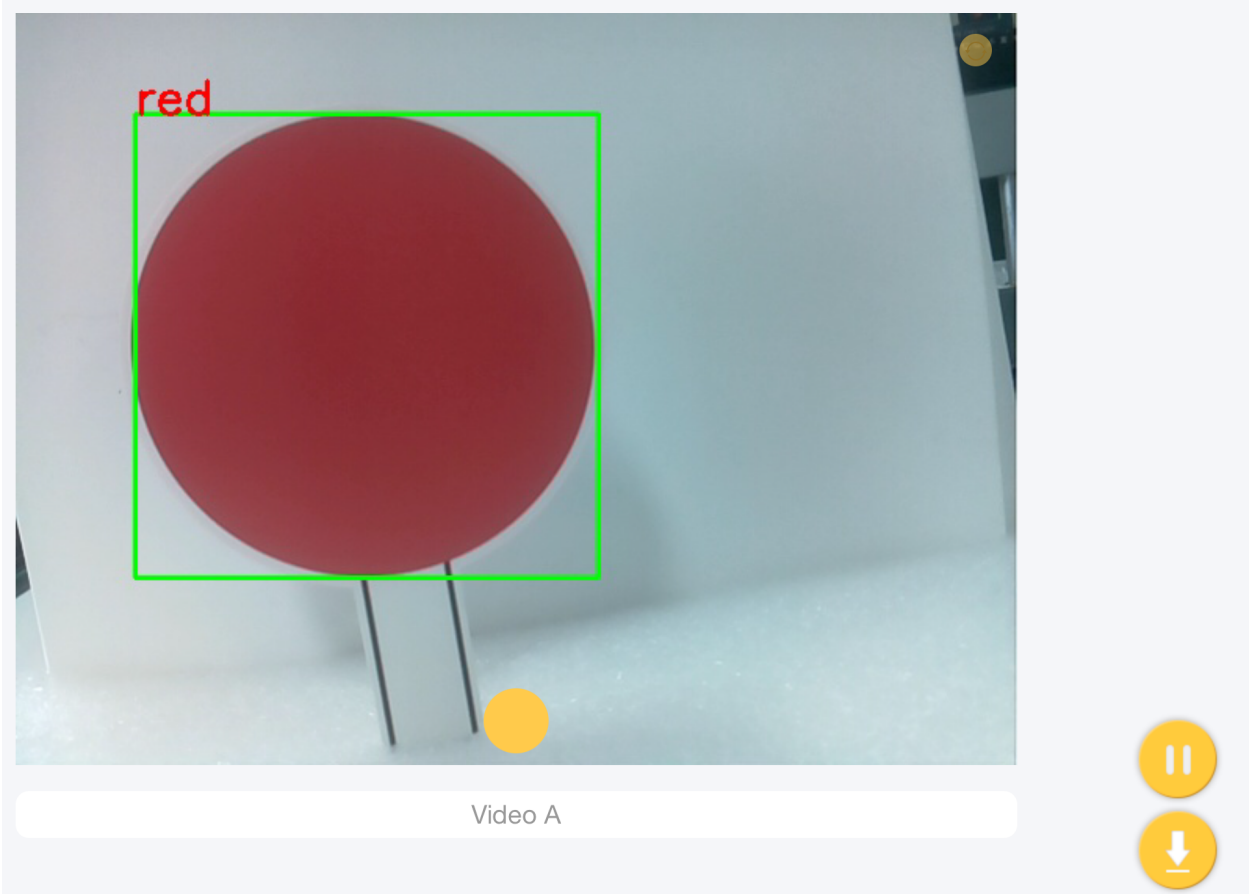


注解： 您可以下载并打印文件 PDF 颜色卡纸来用于颜色检测。

程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。

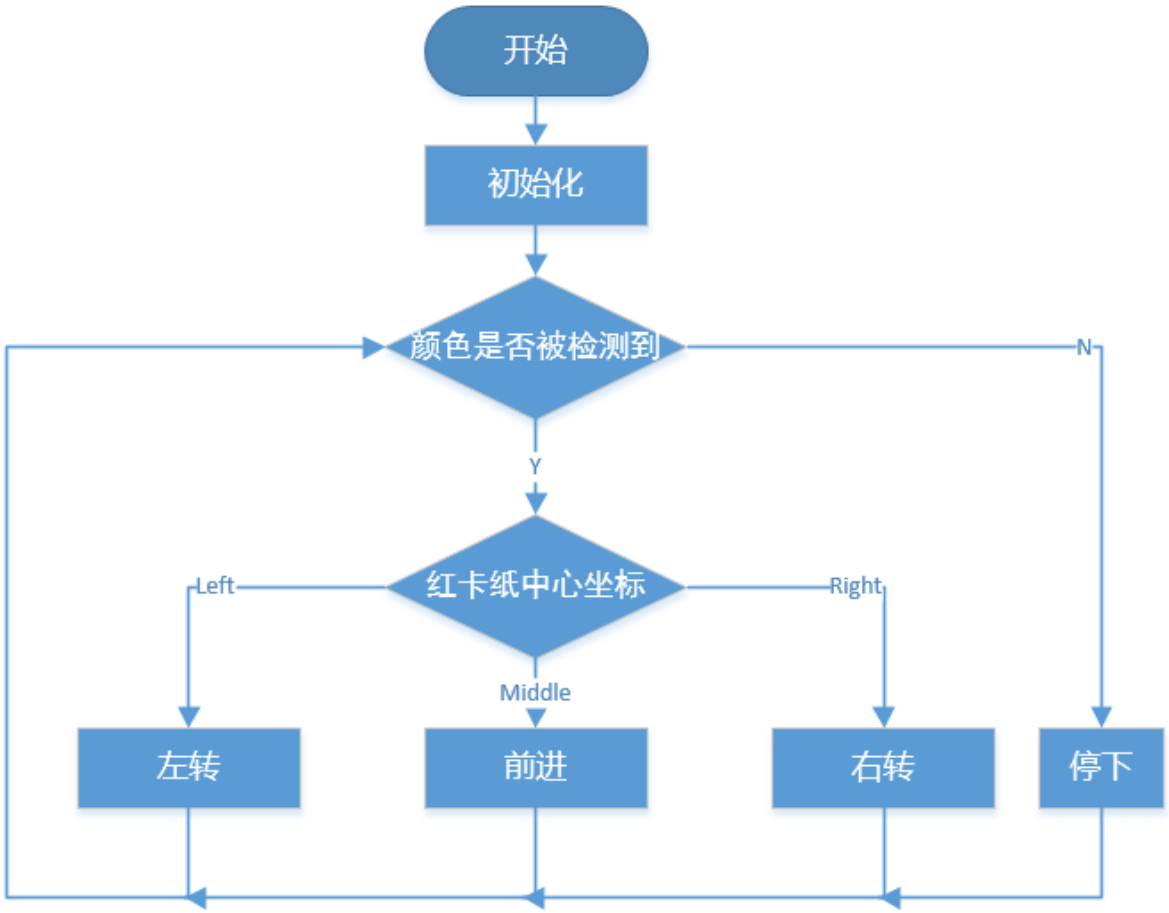
程序运行后，切换到远程控制界面，您将看到以下屏幕，它会将检测到的红色框选出来。若你移动红色卡纸，PiCrawler 也会随着移动。



这个如何运作?

总的来说，这个项目结合了移动, 计算机视觉 和音效 的知识点。

其流程图如下所示：



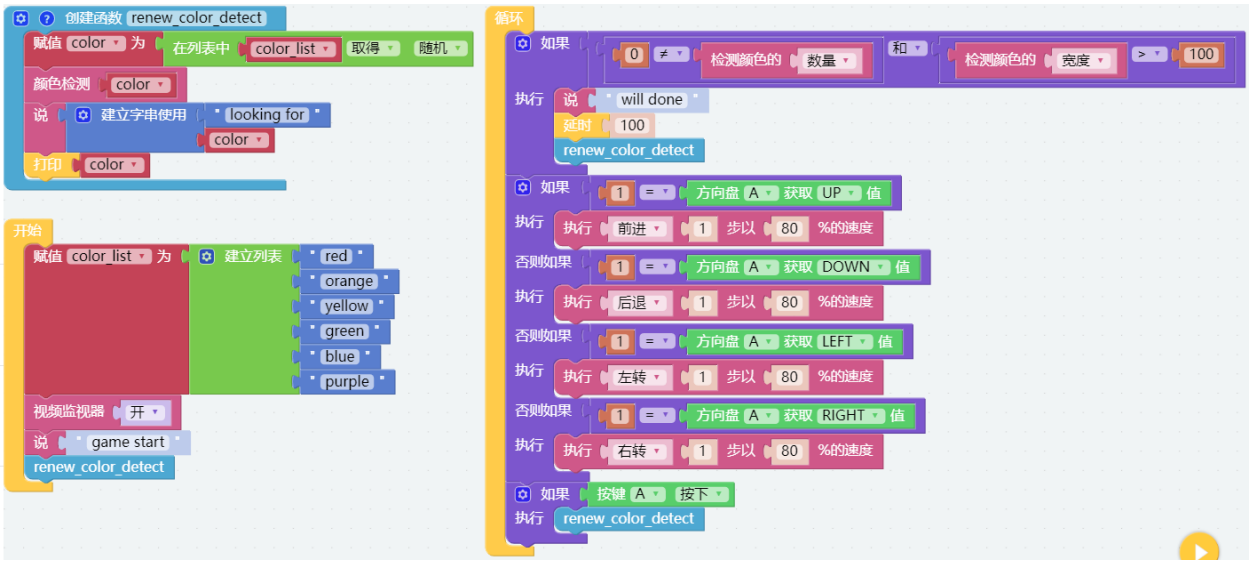
4.9 寻宝

在你的房间里布置一个迷宫，在六个角落放置六张不同颜色的卡片。然后控制 PiCrawler 一一搜索这些卡纸吧！

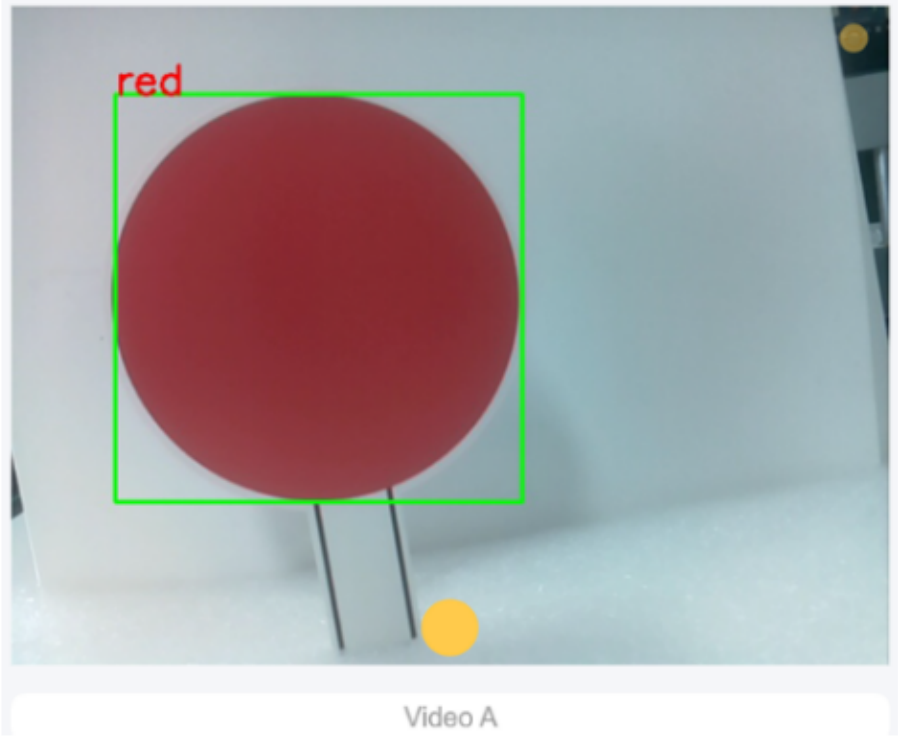
注解： 您可以下载并打印文件 PDF 颜色卡纸来用于颜色检测。

程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。



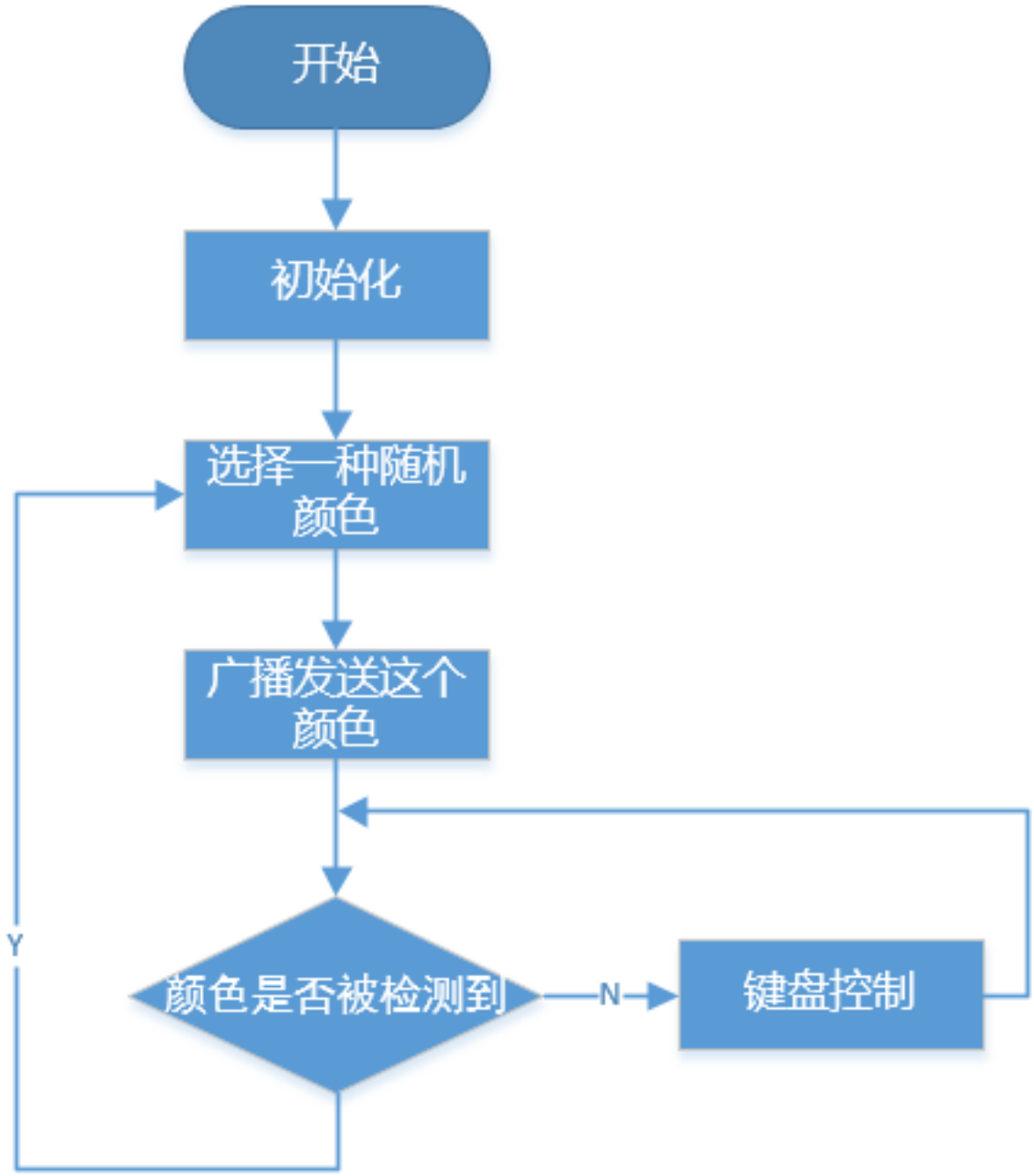
程序运行后，切换到远程控制界面，您将看到以下小部件。你可以点击方向盘 A 来让 PiCrawler 移动，它会将设定的卡片颜色框选出来。你可以按下按键来生成新的颜色指令。



这个如何运作?

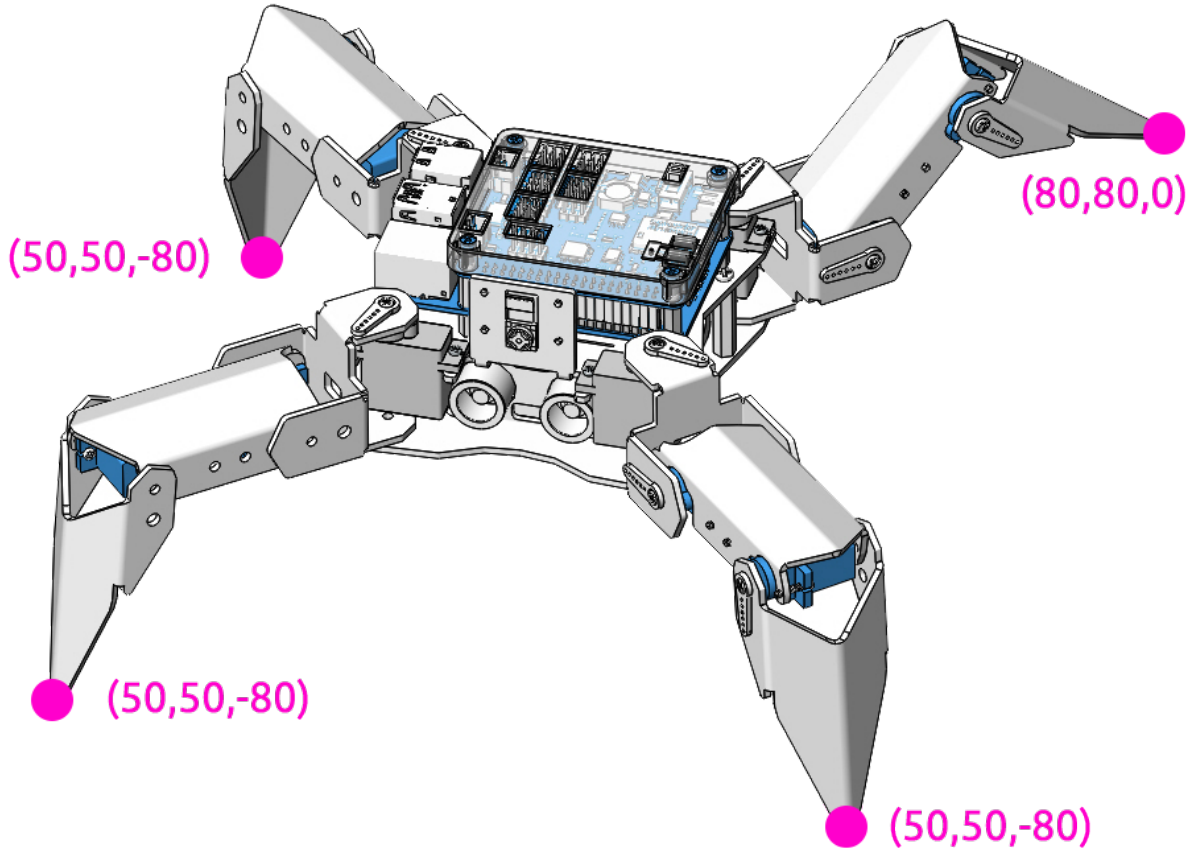
总的来说，这个项目结合了远程控制, 计算机视觉 和音效 的知识点。

其流程图如下所示:


开始
初始化
选择一种随机颜色
广播发送这个颜色
颜色是否被检测到
Y
N
键盘控制

4.10 摆姿势

PiCrawler 可以通过写入坐标数组来呈现特定的姿势。在本项目让它摆出抬起的右后脚姿势。



程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。

程序运行后，PiCrawler 可以呈现特定的姿势。

这个如何运作?

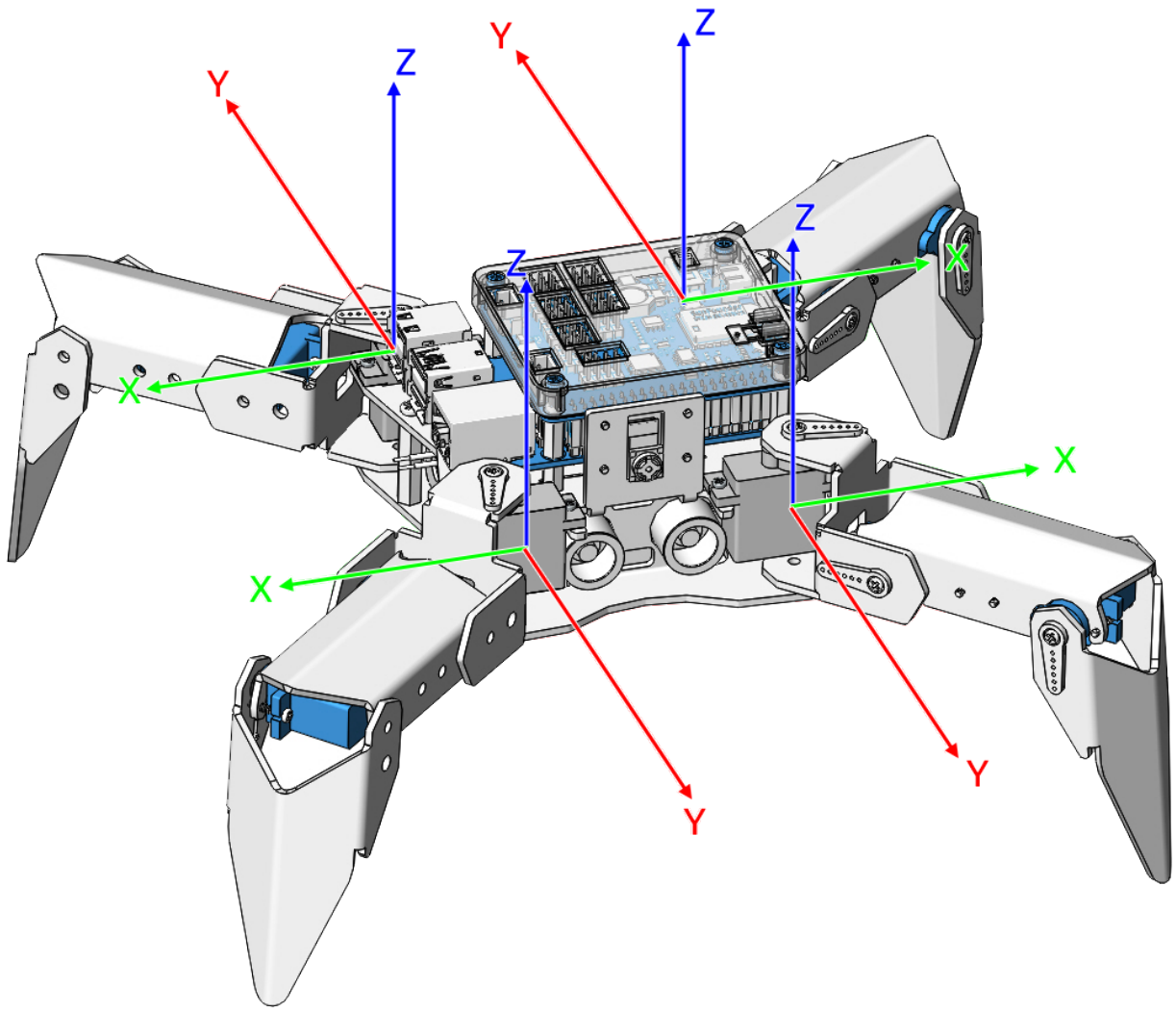
这段代码中，需要注意 **执行单步**积木块。

它有两个用法:

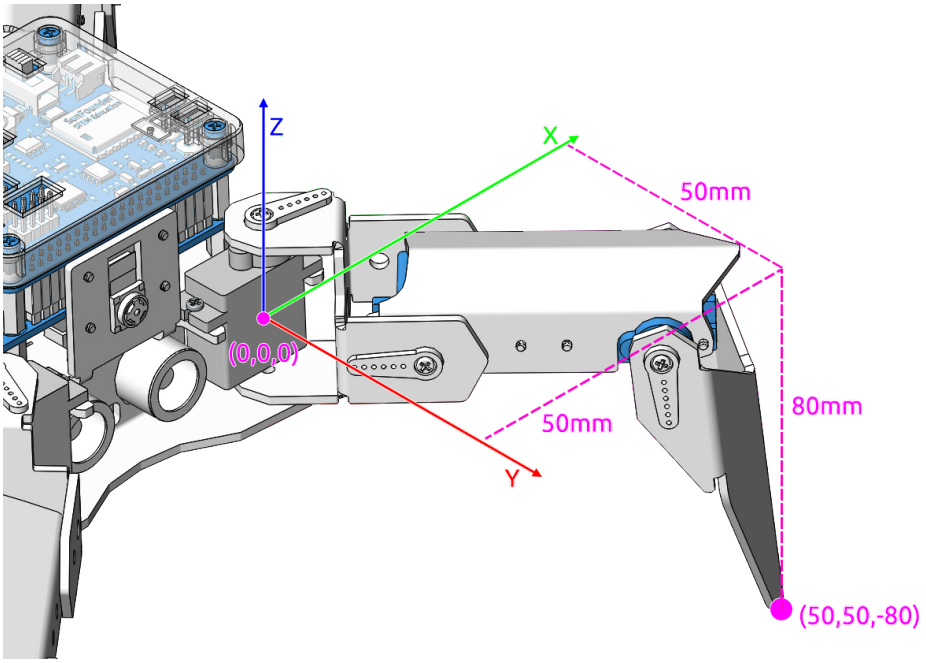
一: 直接用下拉选项选择 **站立**或者 **坐下**。

二: 接收一个自定义的变量，这个变量是一个数组，包含四个空间坐标。

PiCrawler 每只脚都有一个独立的坐标系。如下所示:

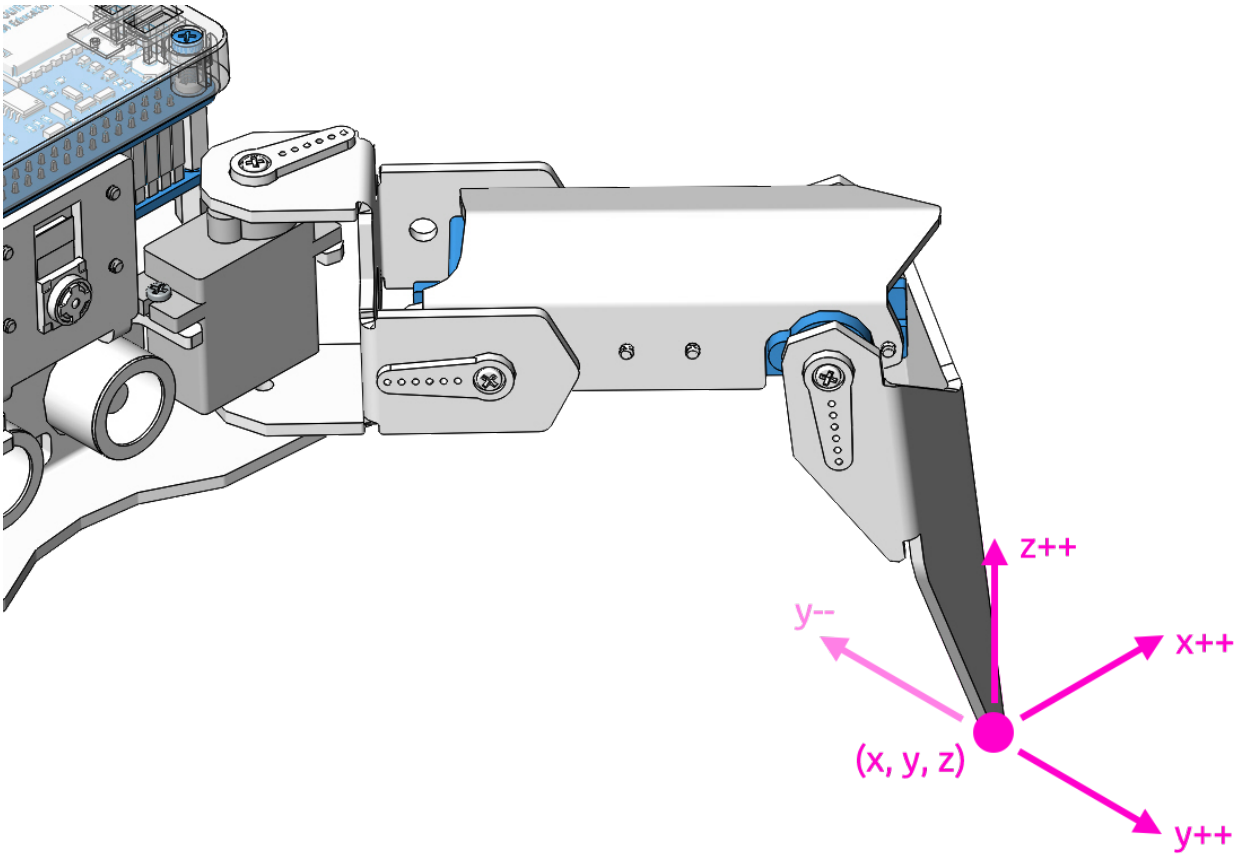


您需要单独测量每个脚趾的坐标。如下所示：



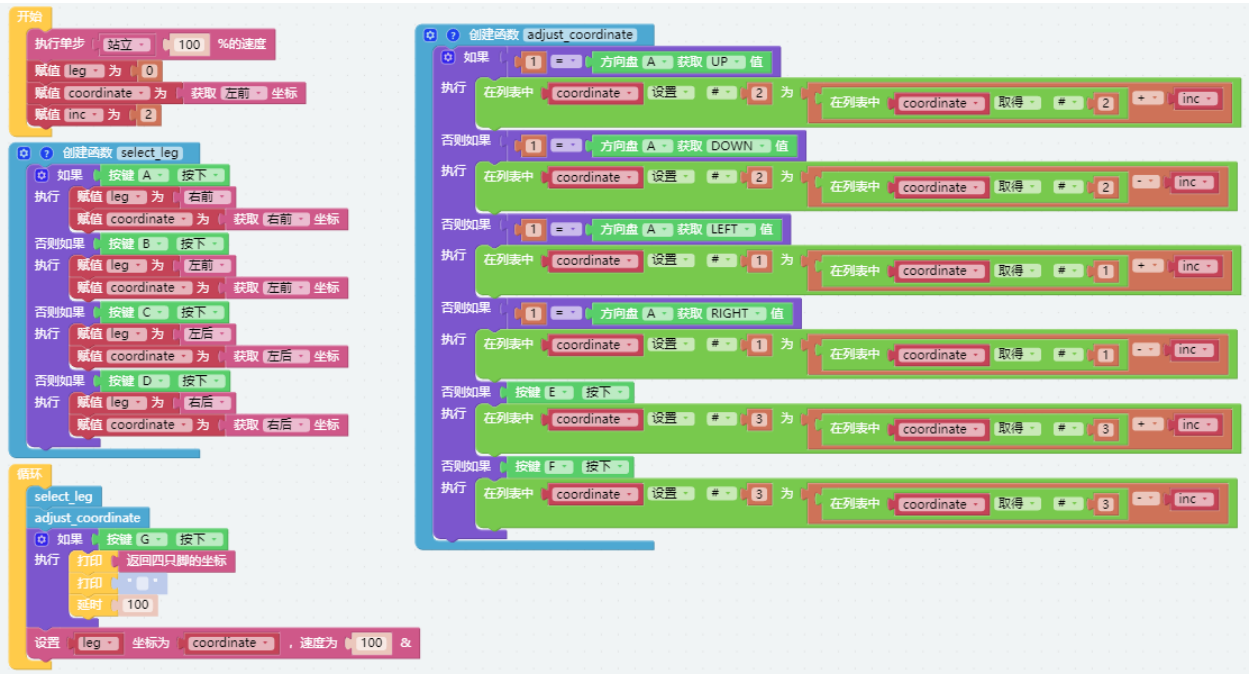
4.11 调整姿势

在本项目中，我们使用远程控制界面的小部件来调整 PiCrawler 的腿来摆出我们想要的姿势。
您可以打印出 PiCrawler 当前动作的坐标值。当您自定义创建独特的动作时，这些坐标值会派上用场。

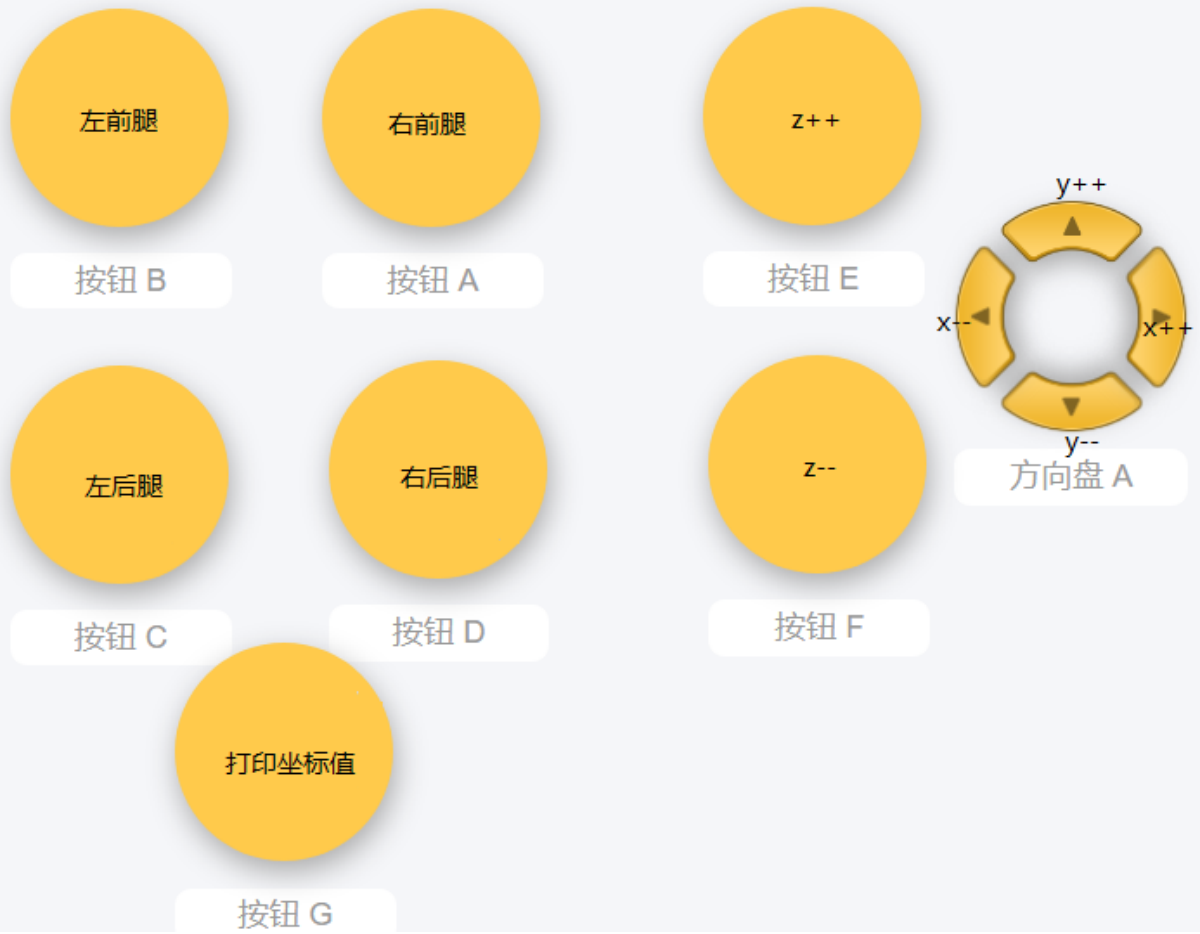


程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。



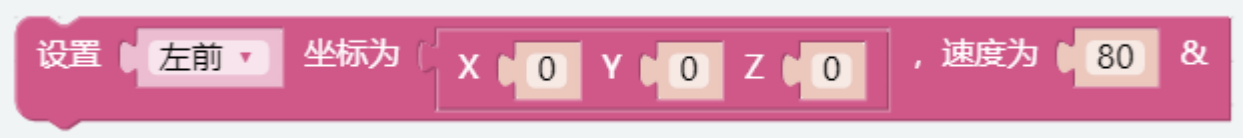
程序运行后，切换到远程控制界面，您将看到以下小部件。



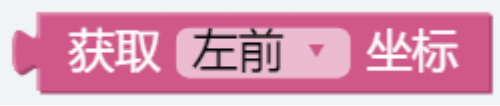
- 按钮 A,B,C 和 D 是用来选择不同的腿。
- 用 E,F 和方向盘用来调整每条腿的 X,Y 和 Z 坐标。
- 将 PiCrawler 调整到自己想要的姿势后，按下按钮 G 来打印所有腿的坐标值。

这个如何运作?

在这个项目中需要注意以下三个积木块:

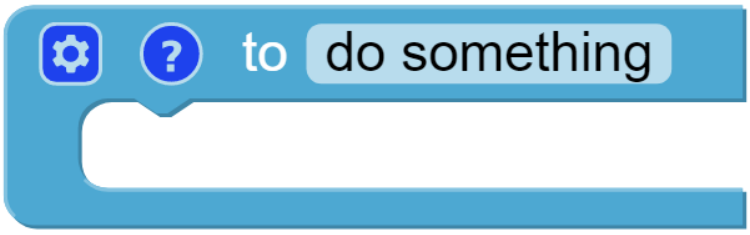


单独修改某条腿的坐标值。



返回相应退的坐标值。

您可能希望使用函数来简化程序，尤其是当您多次执行相同的操作时。把这些操作放到一个新声明的函数中，可以大大简化你的代码。



4.12 记录姿势

我们使用遥控功能控制 PiCrawler 依次摆出几个新的姿势，并记录下这些姿势。然后让 PiCrawler 重复这些姿势。

程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。

程序运行后，切换到远程控制界面，您将看到以下小部件。

- 按钮 A,B,C 和 D 是用来选择不同的腿。
- 用 E,F 和方向盘用来调整每条腿的 X,Y 和 Z 坐标。
- 按键 H 来记录新动作，按键 I 来重复记录的动作。
- 将 PiCrawler 调整到自己想要的姿势后，按下按钮 G 来打印所有腿的坐标值。

这个如何运作?

这个项目可以参考之前的调整姿势。我们在这里新增了记录和回放的功能。
记录功能由以下代码实现。

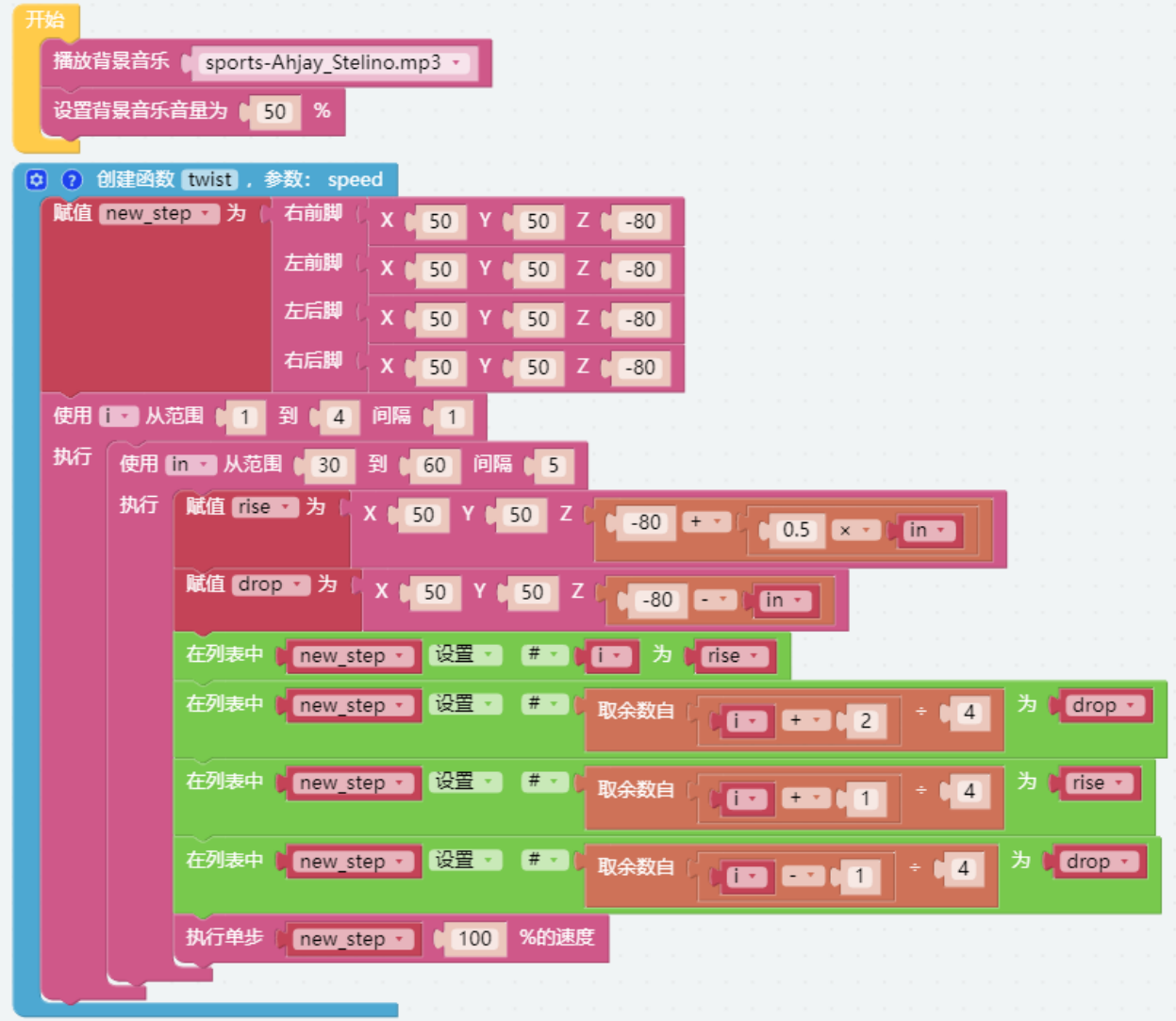
回放功能由以下代码实现。

4.13 组合动作

我们已经知道如何让 PiCrawler 摆出一个特定的姿势，下一步就是将这些姿势组合起来形成一个连续的动作。
这个项目中，我们让 PiCrawler 的四只脚上下摆动，随着音乐跳跃。

程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。


开始
播放背景音乐 sports-Ahjay_Stelino.mp3
设置背景音乐音量为 50 %
创建函数 twist , 参数: speed
赋值 new_step 为 右前脚 X 50 Y 50 Z -80
左前脚 X 50 Y 50 Z -80
左后脚 X 50 Y 50 Z -80
右后脚 X 50 Y 50 Z -80
使用 i 从范围 1 到 4 间隔 1
执行
使用 in 从范围 30 到 60 间隔 5
执行
赋值 rise 为 X 50 Y 50 Z -80 + 0.5 × in
赋值 drop 为 X 50 Y 50 Z -80 - in
在列表中 new_step 设置 # i 为 rise
在列表中 new_step 设置 # 取余数自 i + 2 ÷ 4 为 drop
在列表中 new_step 设置 # 取余数自 i + 1 ÷ 4 为 rise
在列表中 new_step 设置 # 取余数自 i - 1 ÷ 4 为 drop
执行单步 new_step 100 %的速度

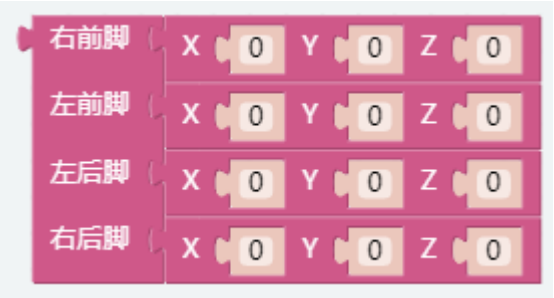
程序运行后，PiCrawler 的四只脚上下摆动，随着音乐跳跃。

这个如何运作?

我们用两层循环使坐标数组 `new_step` 产生连续有规律的变化，同时用 **执行单步...** 来执行 `new_step` 里的姿势从而形成连续的动作。

您可以从调整姿势 课程中直观的获取每个姿势对应的坐标值数组。

坐标值数组如下图所示：



它本质上是一个二维数组,可以被列表类中的积木块处理。它的结构是 `[[right front],[left front],[left rear],[right rear]]` ，在这个例子中对应于右前方，左前方，左后方和右后方。

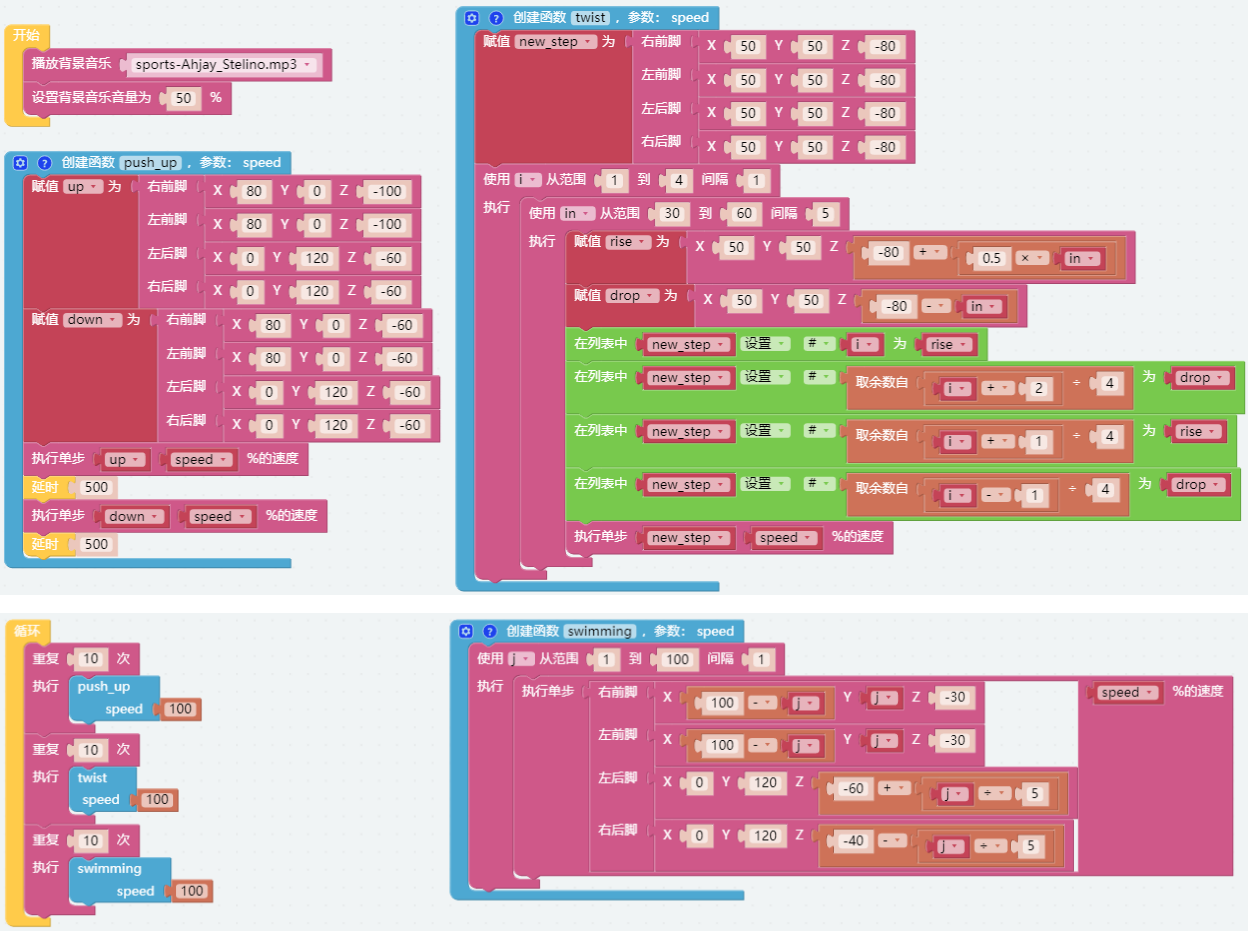
4.14 情感机器人

这个例子展示了 PiCrawler 的几个有趣的自定义动作。

可以把这个项目当成组合动作 项目的补充。

程序

注解： 你可以直接打开我们提供的示例或者是按照下图来编写程序，详细教程请参考打开或新建项目。



程序运行后，PiCrawler 将随着音乐做不同的动作。

CHAPTER 5

附录

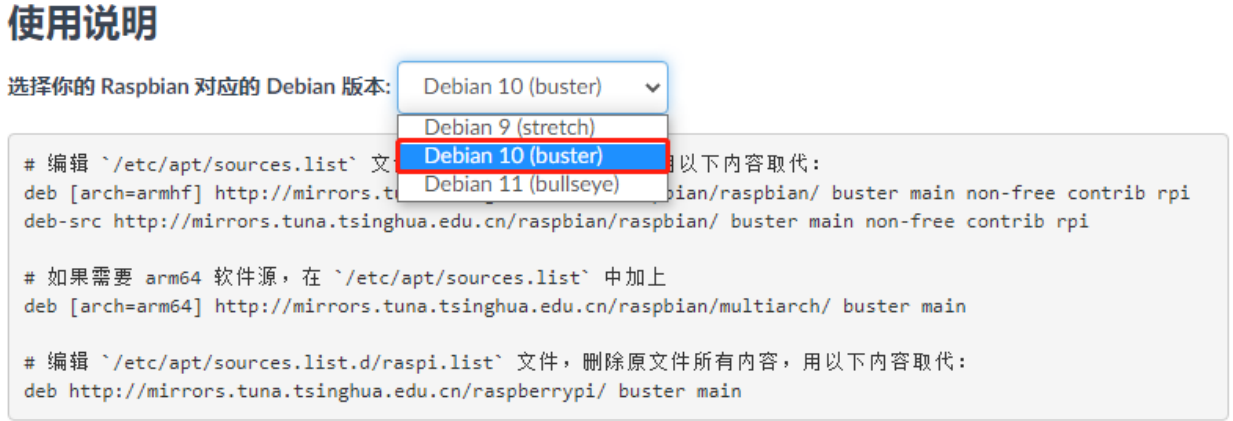
5.1 apt 和 pip 更换国内源

树莓派系统默认的 apt 源和 pip 源都是国外的服务器，使用国内网络访问可能会发生超时（ReadTimeoutErro），或被拒绝访问的情况，如果是这样我们可以将 apt 和 pip 更改为国内的源，步骤如下所示：

1.apt 更换国内源

1）访问链接：https://mirrors.tuna.tsinghua.edu.cn/help/raspbian/ 以了解配置文件修改详情。

2）不同的树莓派系统版本修改不同，先选择对应的版本，比如我的是 Debian 10 （buster），如果你的树莓派系统是 bullseye 则选择对应版本，若没有符合的版本则请重新安装 buster 及以下版本的系统。



注解： 一般不删除原内容，可以将其用 # 注释掉。

3）我们这里以 raspbian buster 版本为例，用 nano 命令打开 `/etc/apt/sources.list` 文件。

```
sudo nano /etc/apt/sources.list
```

4）然后用 # 将原本的内容注释掉，在最后面附上下面的代码。

```
# deb http://raspbian.raspberrypi.org/raspbian/ buster main contrib non-free rpi
# Uncomment line below then 'apt-get update' to enable 'apt-get source'
#deb-src http://raspbian.raspberrypi.org/raspbian/ buster main contrib non-free rpi

deb [arch=armhf] http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ buster main␣
↪non-free contrib rpi
deb-src http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ buster main non-free␣
↪contrib rpi
```

5）按下 Ctrl+O 保存，按下 Ctrl+X 和 Y 退出。

6）用 nano 命令打开 etc/apt/sources.list.d/raspi.list 文件。

```
sudo nano /etc/apt/sources.list.d/raspi.list
```

7）然后用 # 将原本的内容注释掉，在最后面附上 deb... 代码。

```
# deb http://archive.raspberrypi.org/debian/ buster main
# Uncomment line below then 'apt-get update' to enable 'apt-get source'
#deb-src http://archive.raspberrypi.org/debian/ buster main

deb http://mirrors.tuna.tsinghua.edu.cn/raspberrypi/ buster main ui
```

8）按下 Ctrl+O 保存，按下 Ctrl+X 和 Y 退出。

9）用以下命令更新软件列表:

```
sudo apt update
```

2.(Pypi) pip 更换国内源

可以参考链接 https://mirrors.tuna.tsinghua.edu.cn/help/pypi/ 了解详情。

有两种方法修改配置文件。

方法一：使用 pip 指令设置

```
pip3 config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
```

方法二：手动编辑文件

Linux/Mac os 环境中，配置文件位置在 ~/.pip/pip.conf（如果不存在则手动创建该目录和文件）。

```
sudo mkdir -p ~/.pip
sudo nano ~/.pip/pip.conf
```

然后按如下编辑文件内容

```
[global]
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
[install]
trusted-host = https://pypi.tuna.tsinghua.edu.cn
```

最后 apt 和 pip 都已经更换了国内下载源，这样下载速度就会提高很快不会导致下载失败了。

5.2 Filezilla 软件

文件传输协议 (FTP) 是一种标准通信协议，用于在计算机网络上将计算机文件从服务器传输到客户端。

Filezilla 是一款开源软件，不仅支持 FTP，还支持 FTP over TLS (FTPS) 和 SFTP。我们可以使用 Filezilla 将本地文件（如图片和音频等）上传到树莓派，或者从树莓派下载文件到本地。

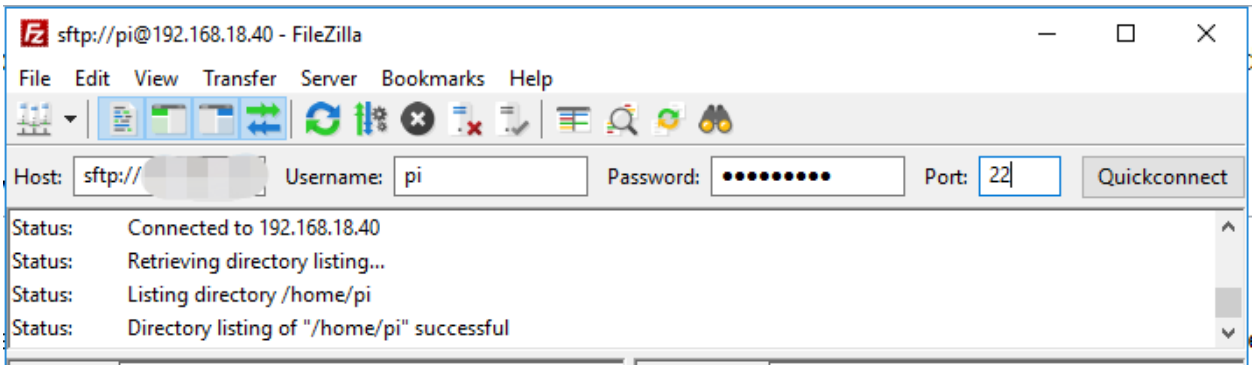
第 1 步: 下载 Filezilla。

在 Filezilla 官方网站 下载客户端, Filezilla 有一个很好的教程，请参考: Filezilla 文档 。

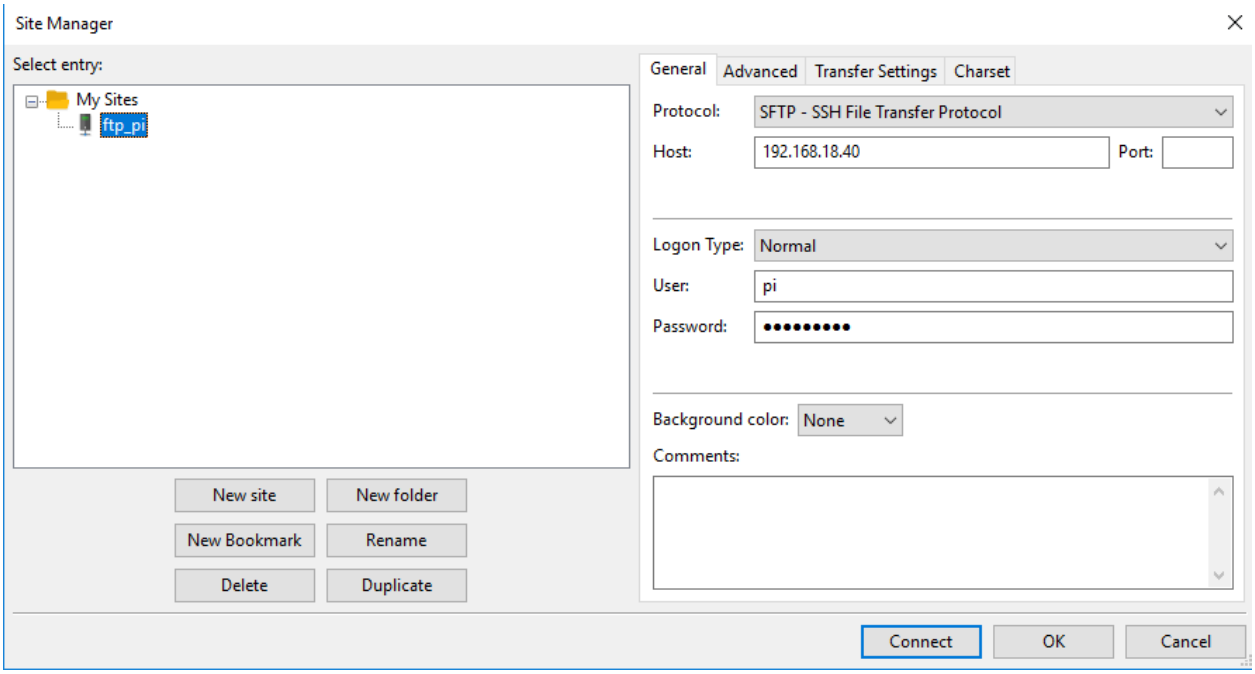
第 2 步: 连接树莓派

快速安装后打开它，现在 连接到 FTP 服务器。

它有 3 种连接方式，这里我们使用 **快速连接**栏。输入 **主机名/IP** 、**用户名**、**密码**和 **端口（22）**，然后点击 **快速连接**或按 **回车**连接到服务器。

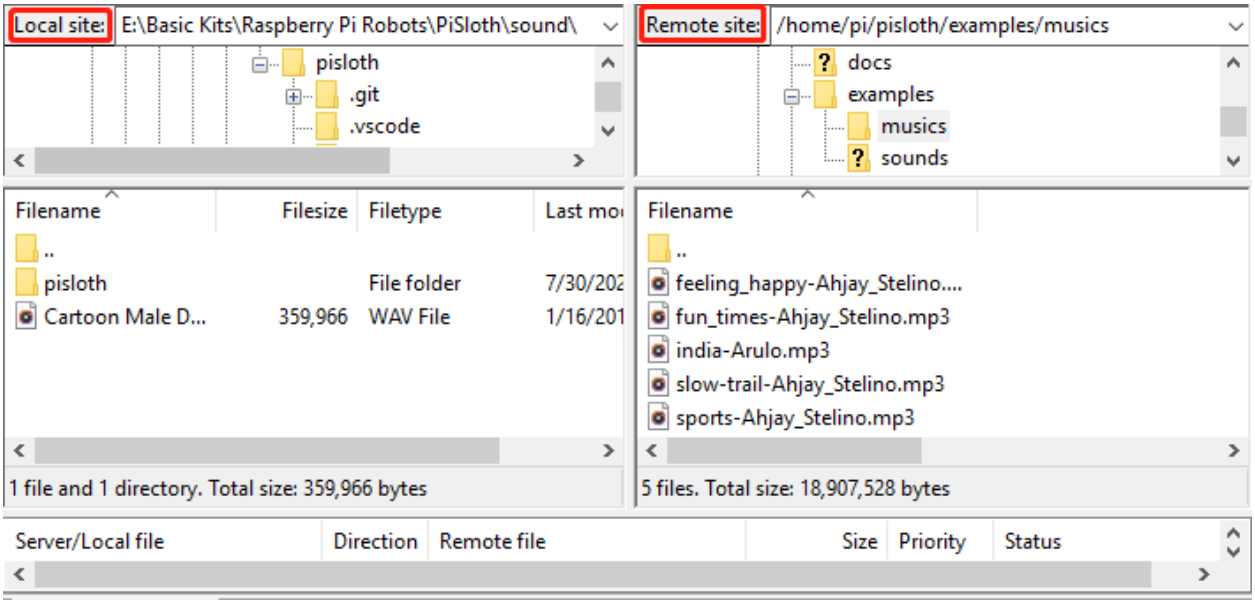


注解: 快速连接是测试您的登录信息的好方法。如果要创建永久条目，可以在快速连接成功后选择 **File-> Copy Current Connection to Site Manager** （文件-> 将当前连接复制到站点管理器），输入名称并单击 **确定**。下次您将能够通过在 **File -> Site Manager** (文件-> 站点管理器。) 中选择先前保存的站点进行连接。



第 3 步: 上传/下载文件。

您可以通过拖放将本地文件上传到树莓派，也可以将树莓派文件中的文件下载到本地。

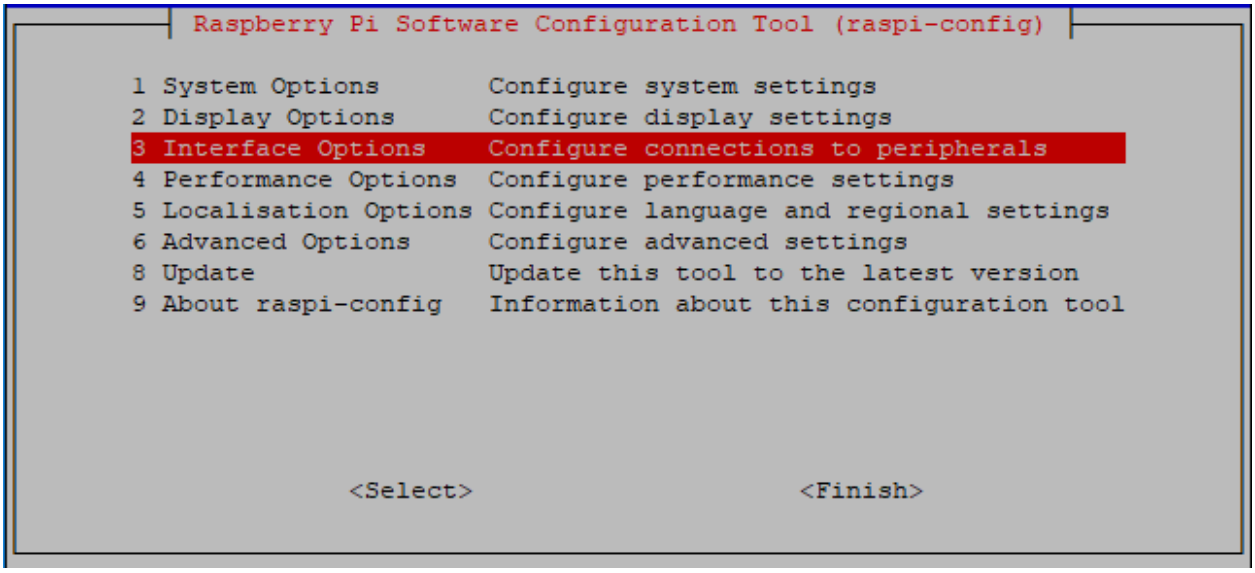


5.3 I2C 配置

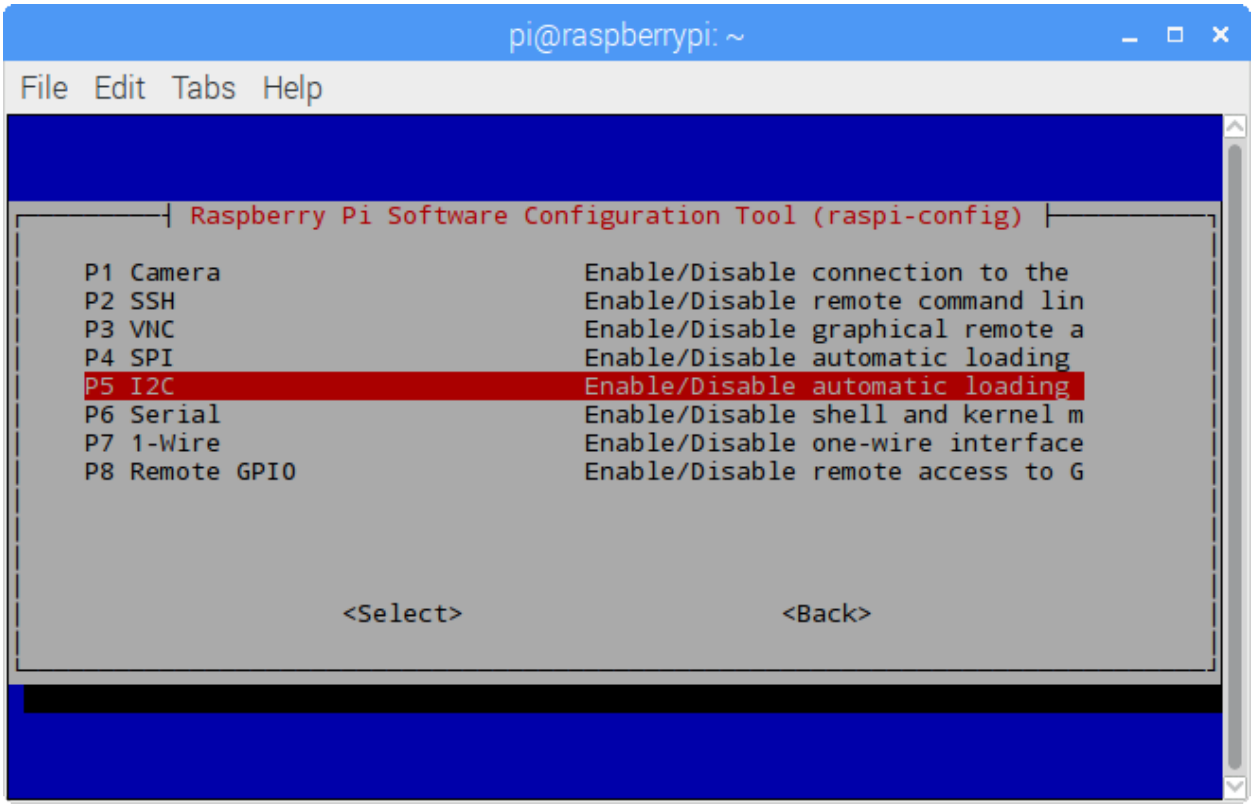
启用你的树莓派的 I2C 端口。如果你已经启用了它，请跳过这一步；如果你不知道你是否已经做了，请继续。

```
sudo raspi-config
```

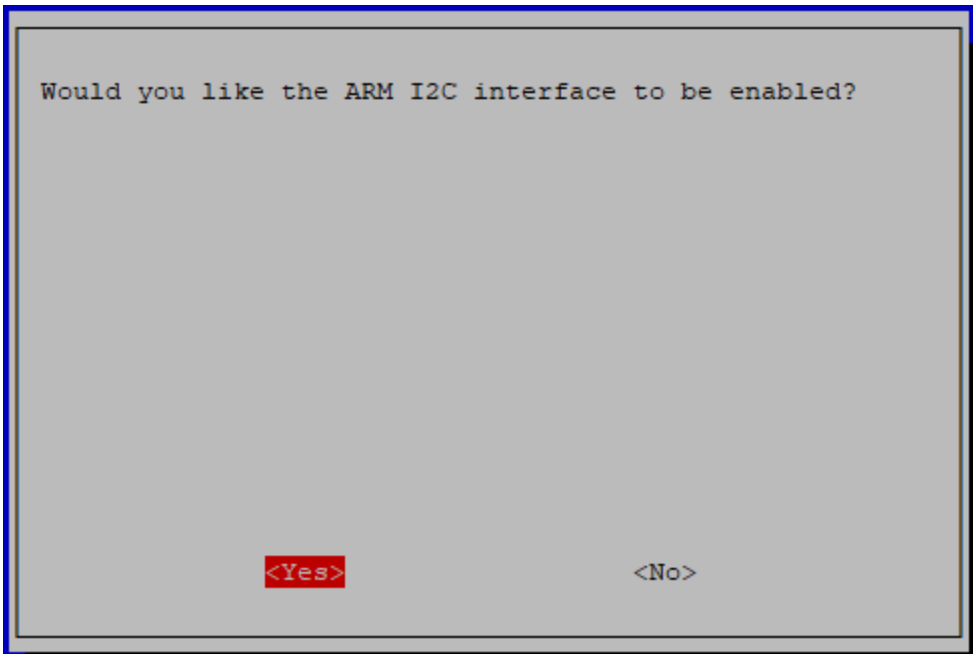
3 Interfacing options



P5 I2C



<Yes>, 然后 <Ok> -> <Finish>.



5.4 远程桌面

有两种方法可以远程控制树莓派的桌面。

VNC 和 **XRDP**，你可以使用它们中的任何一种。

5.4.1 VNC

你可以通过 VNC 使用远程桌面的功能。

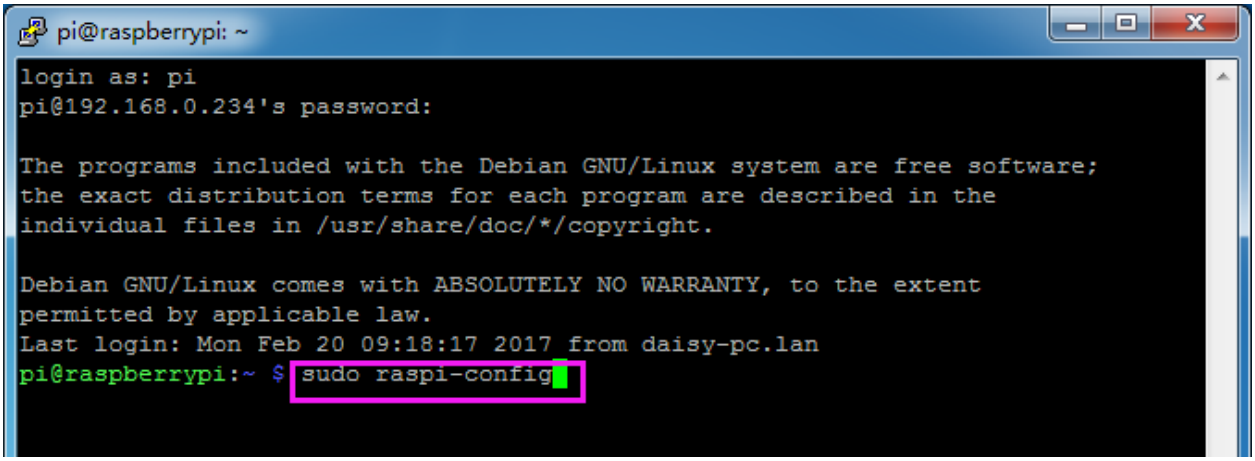
启用 VNC 服务

系统中已经安装了 VNC 服务。默认情况下，VNC 被禁用。默认情况下，VNC 是禁用的。你需要在配置中启用它。

第 1 步

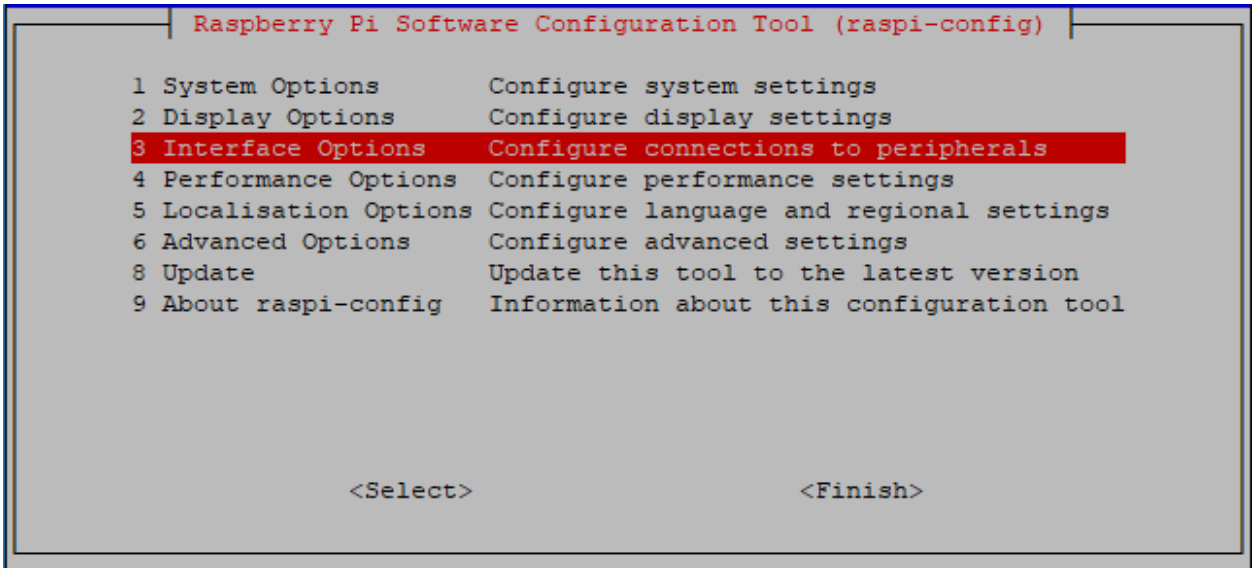
输入以下命令。

```
sudo raspi-config
```



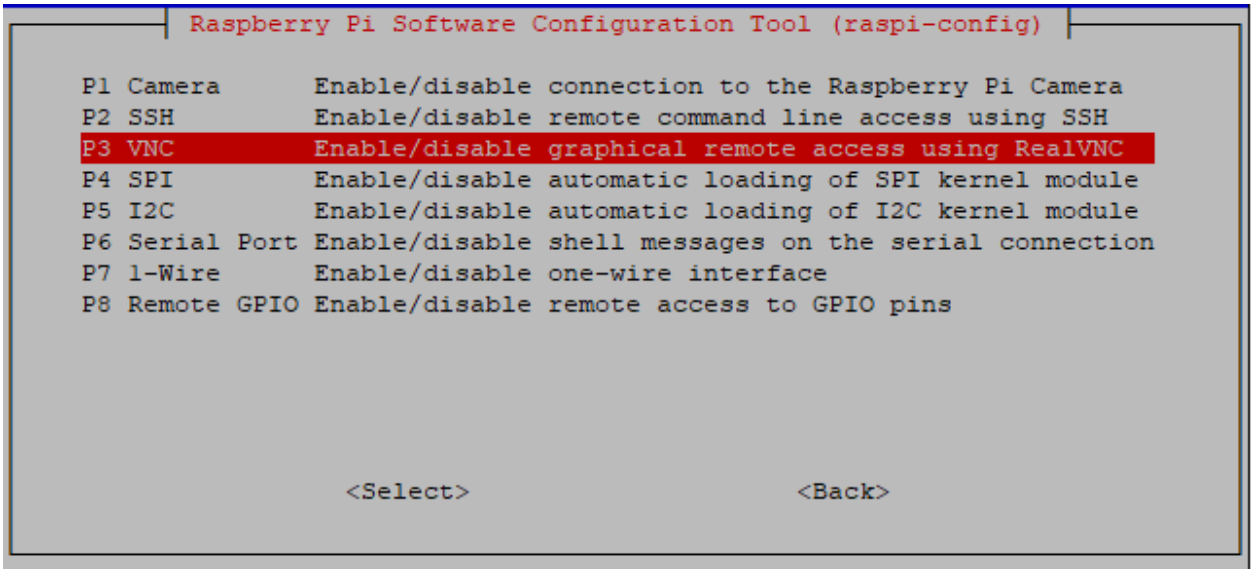
第 2 步

选择 **3 Interfacing Option**，按你键盘上的向下箭头键。键盘上的向下箭头键，然后按 **Enter** 键。



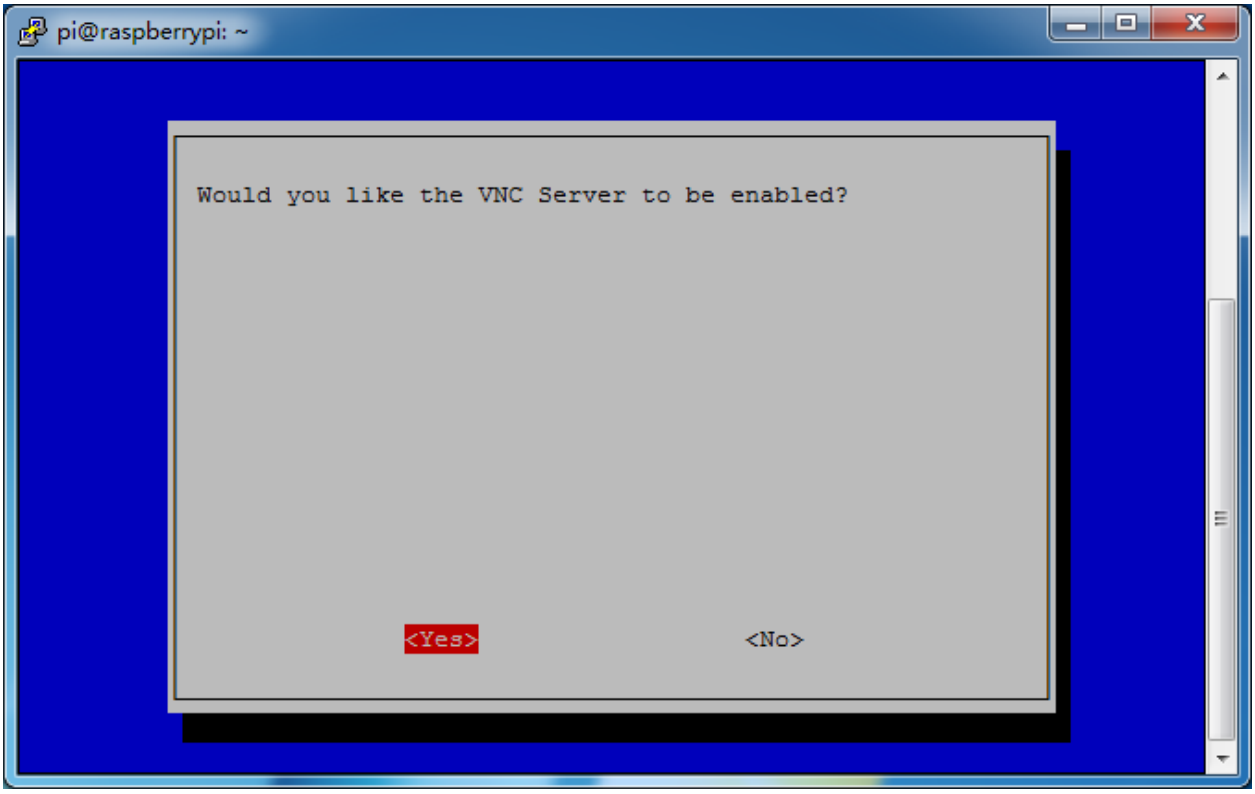
第 3 步

P3 VNC



第 4 步

选择 **Yes -> OK -> Finish** 来退出配置。



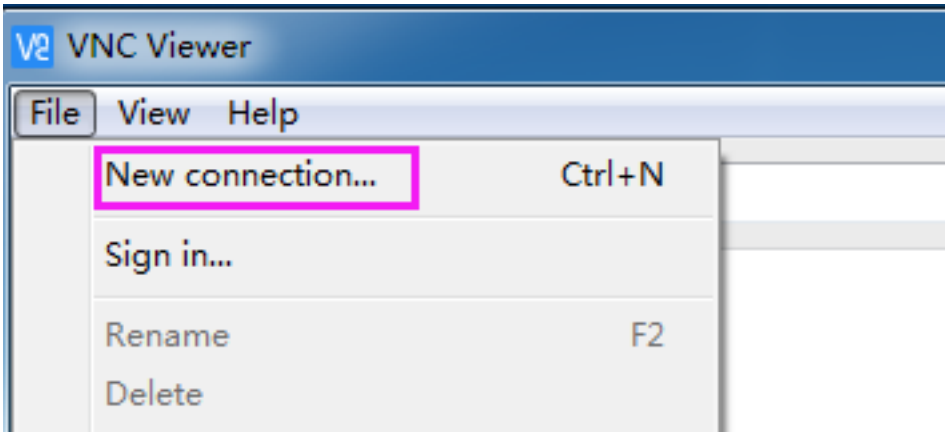
登录 VNC

第 1 步

你需要在个人电脑上下载并安装 VNC Viewer。安装完成后，打开它。

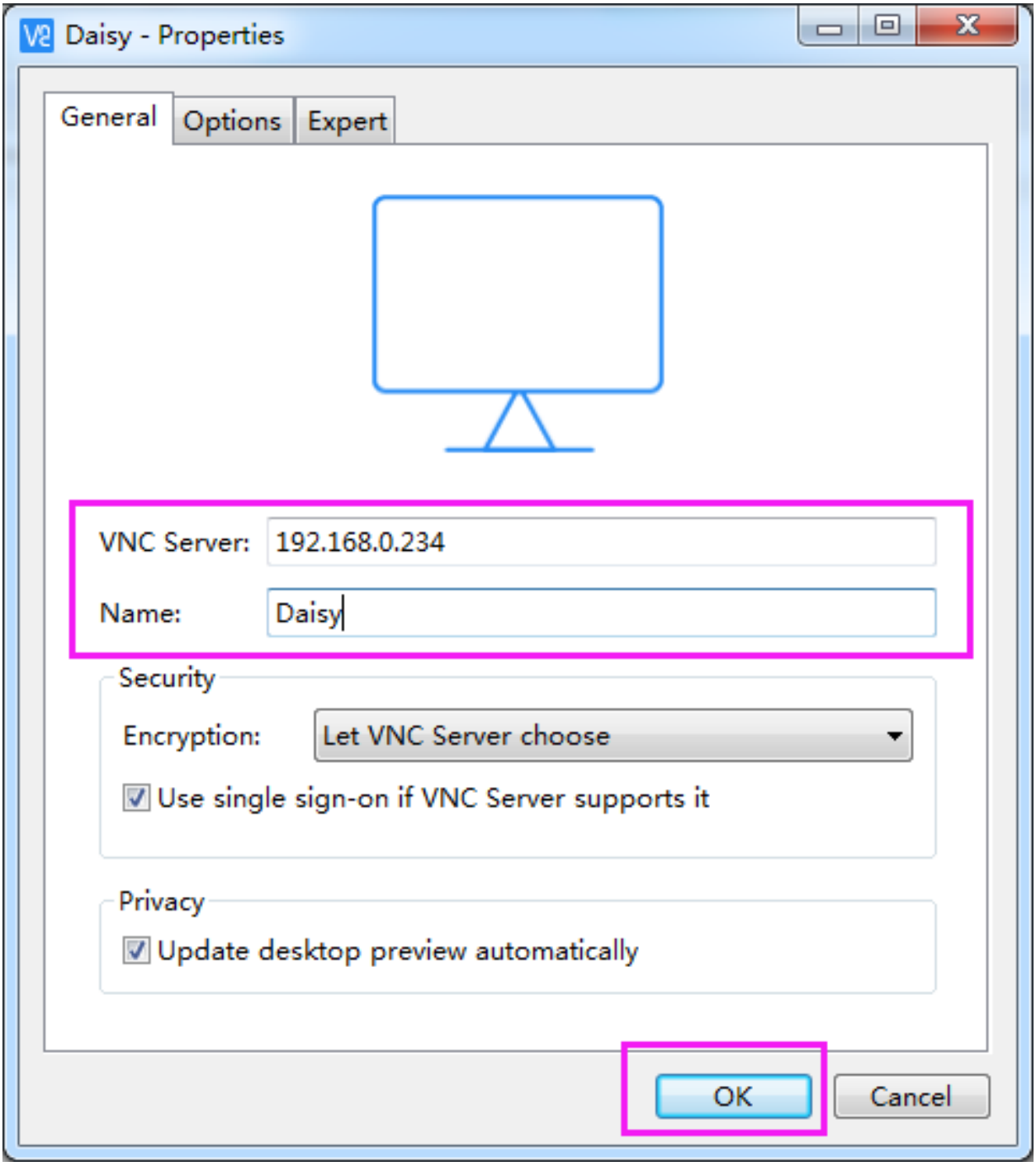
第 2 步

然后选择”**New connection**”.



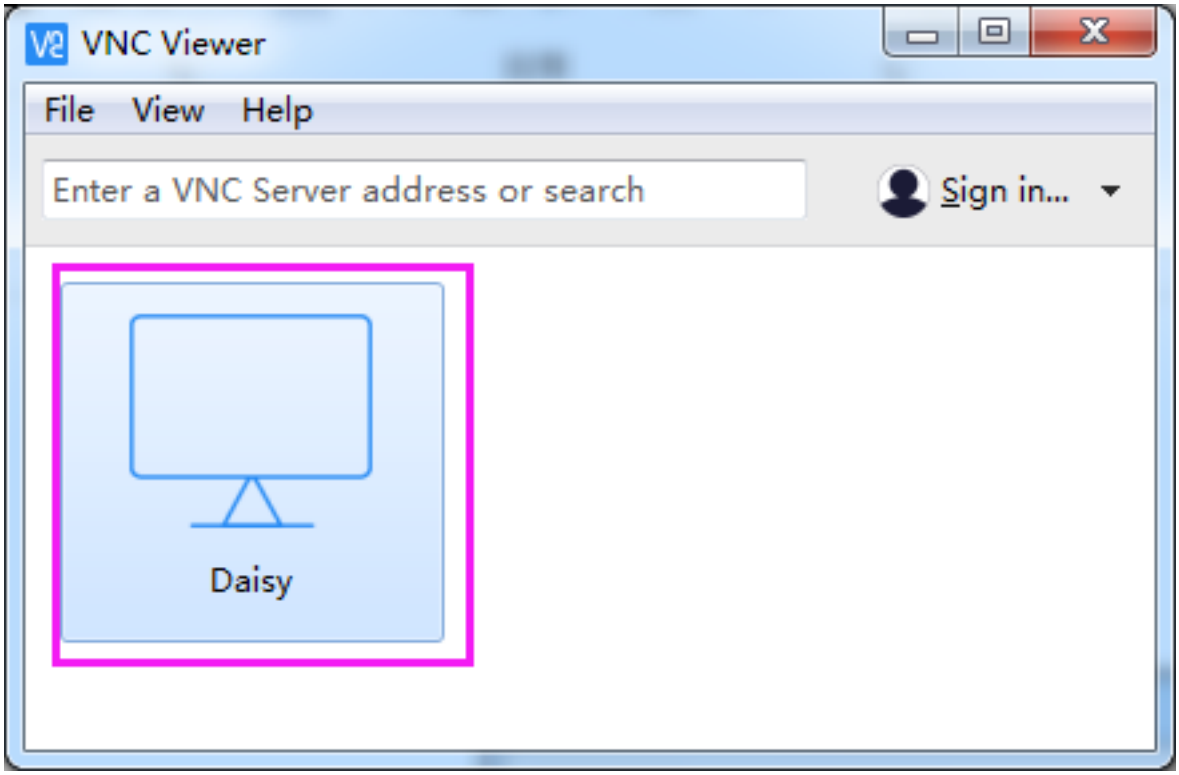
第 3 步

输入树莓派的 IP 地址和任意 **名称**。



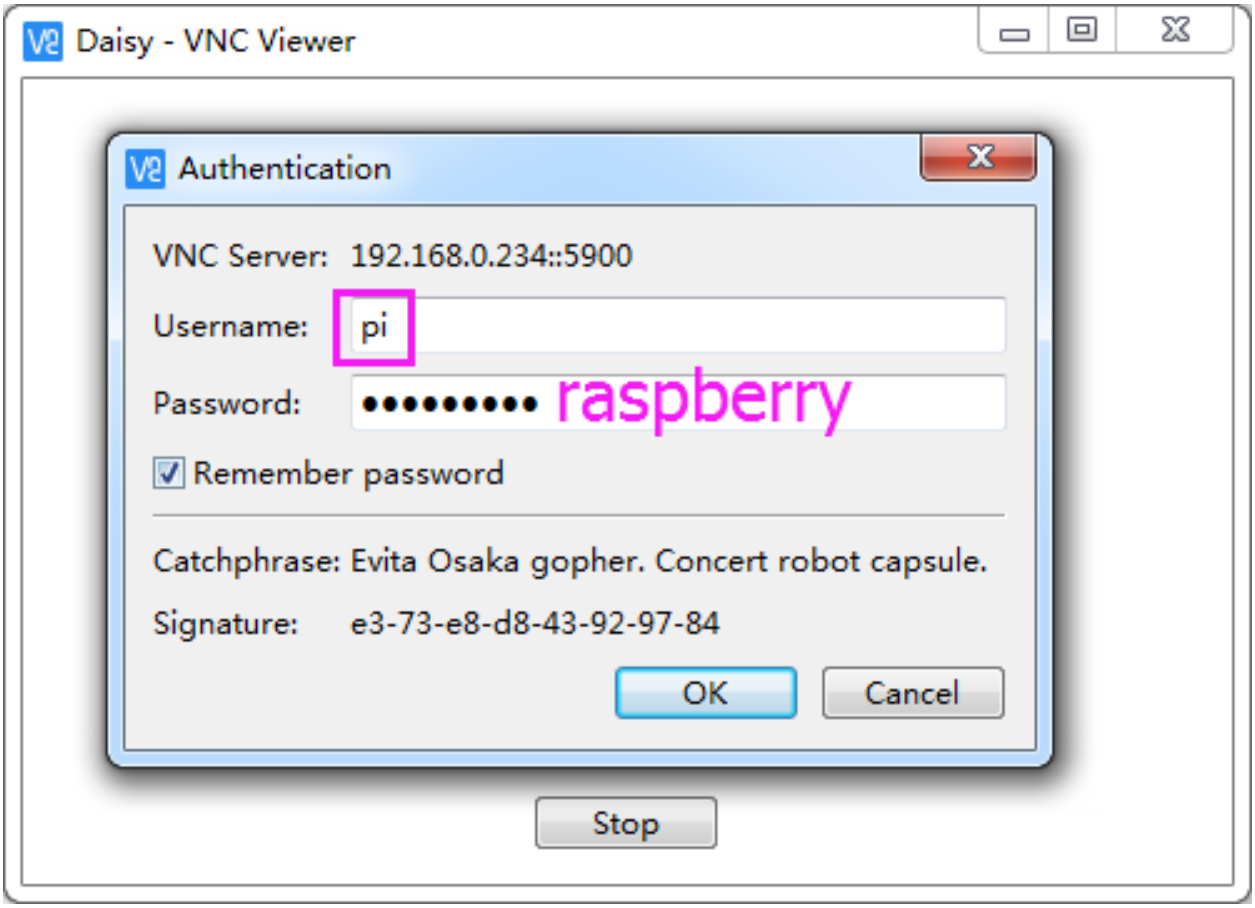
第 4 步

双击刚刚创建的 **连接**。



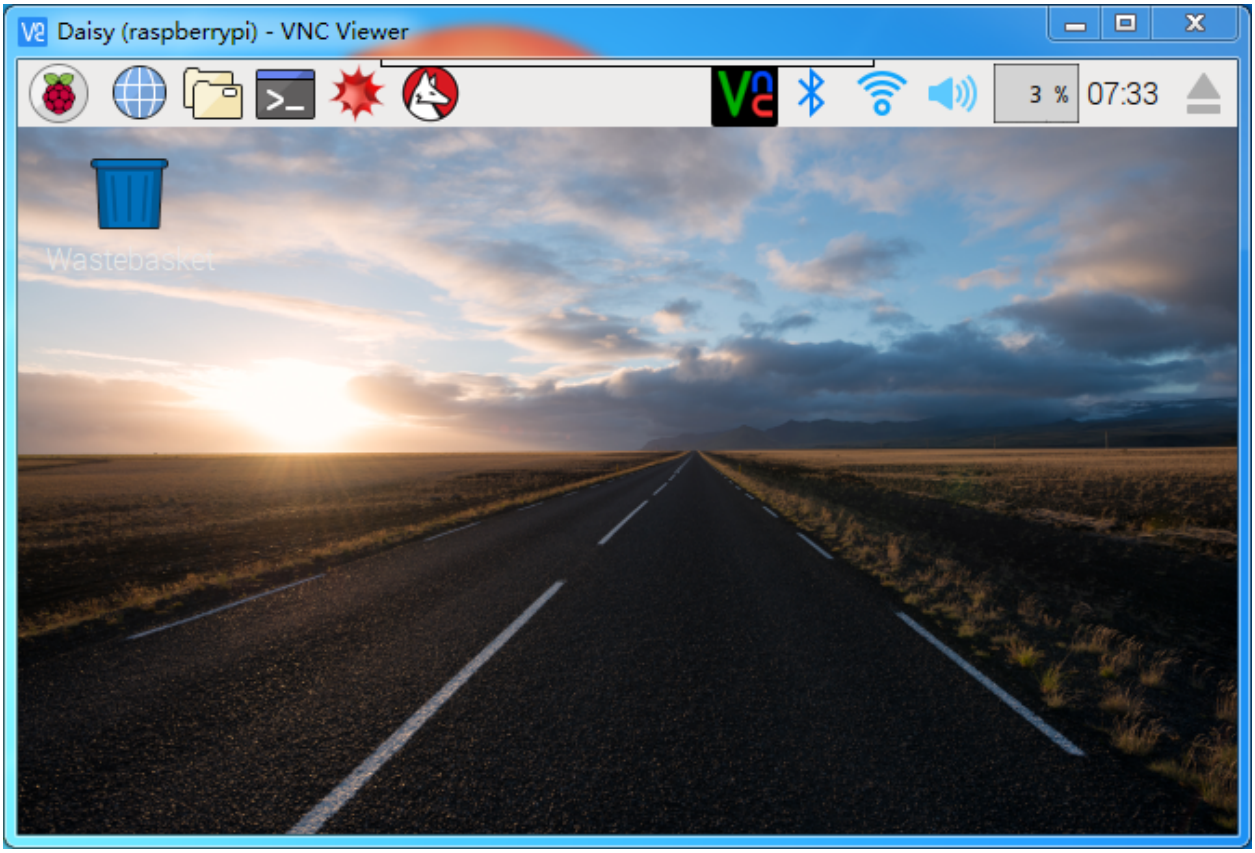
第 5 步

输入用户名（**pi**）和密码（默认为 **raspberry**）。



第 6 步

现在你可以看到树莓派的桌面了。



VNC 部分就到此为止了。

5.4.2 XRDP

另一种远程桌面的方法是 XRDP，它使用 RDP（微软远程桌面协议）提供图形化登录到远程机器。远程桌面协议）。

安装 XRDP

第 1 步

通过使用 SSH 登录到树莓派。

第 2 步

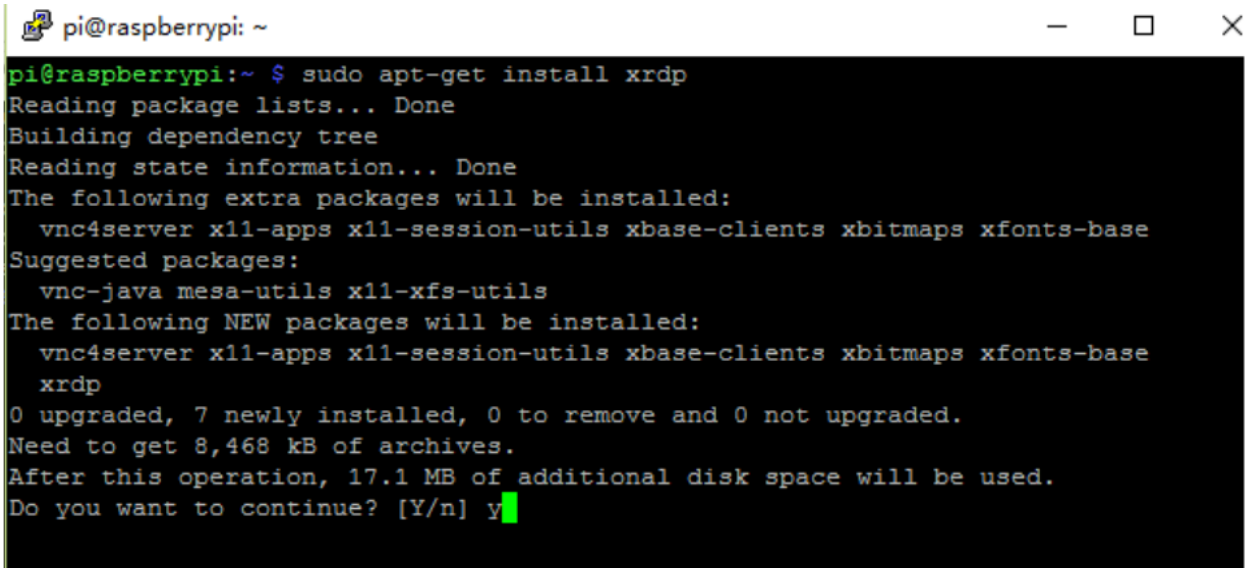
输入以下说明来安装 XRDP。

```
sudo apt-get update
sudo apt-get install xrdp
```

第 3 步

后来，安装开始了。

按（”Y”），然后按（”Enter”）来确认。



第 4 步

安装完成后，你应该使用 Windows 远程桌面应用程序登录到你的树莓派。使用 Windows 远程桌面应用程序。

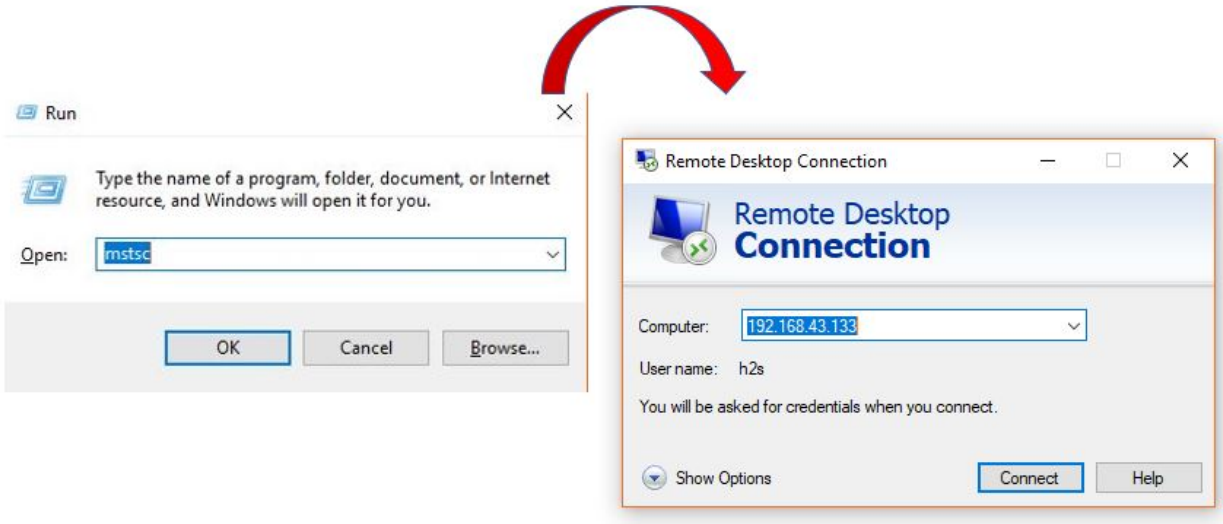
登录到 XRDP

第 1 步

如果你是一个 Windows 用户，你可以使用 Windows 自带的远程桌面功能。自带的远程桌面功能。如果你是 Mac 用户，你可以从 APP Store 下载并使用微软远程桌面，两者之间没有太大区别。两者之间没有什么区别。接下来的例子是 Windows 远程桌面。

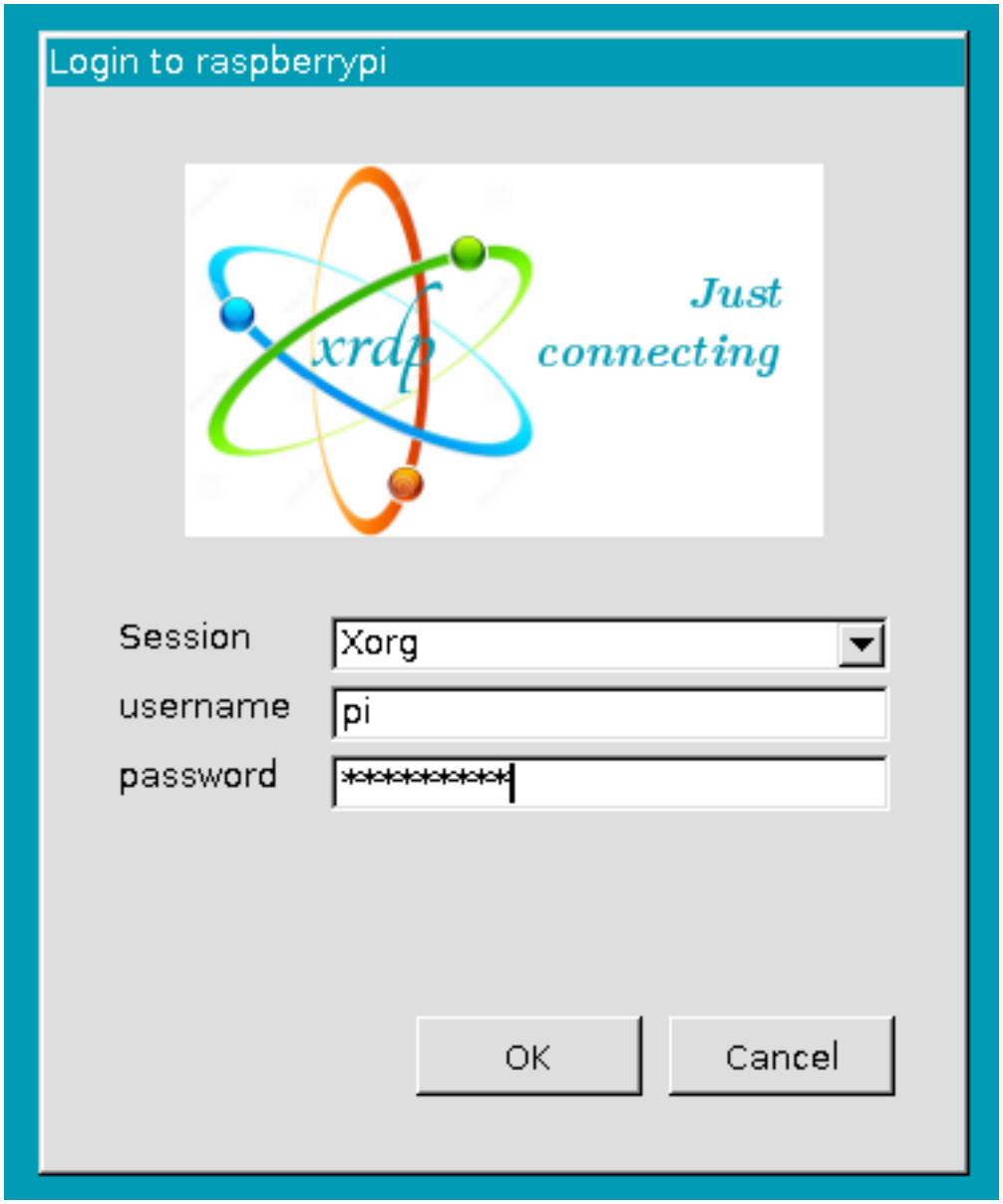
第 2 步

在运行（WIN+R）中键入”**mstsc**”，打开远程桌面连接，并输入树莓派的 IP 地址，然后点击 Connect”。



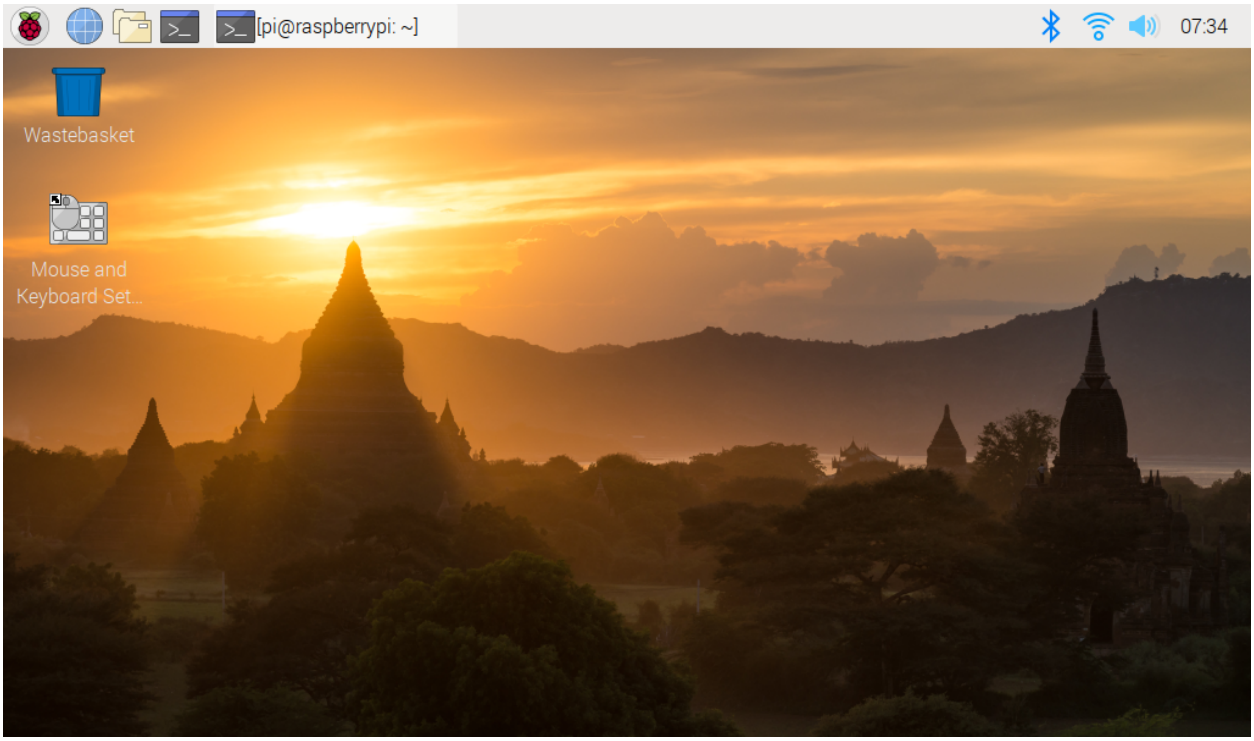
第 3 步

然后弹出 xrdp 登录页面。请键入您的用户名和密码。之后，请点击”OK”。在你第一次登录的时候。你的用户名是”pi”，密码是”raspberry”。



第 4 步

在这里，你通过使用远程桌面成功登录到 RPi。



5.5 关于电池

适用参数

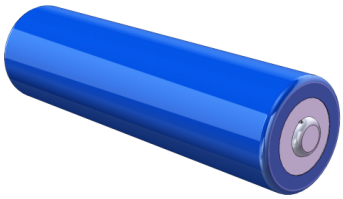
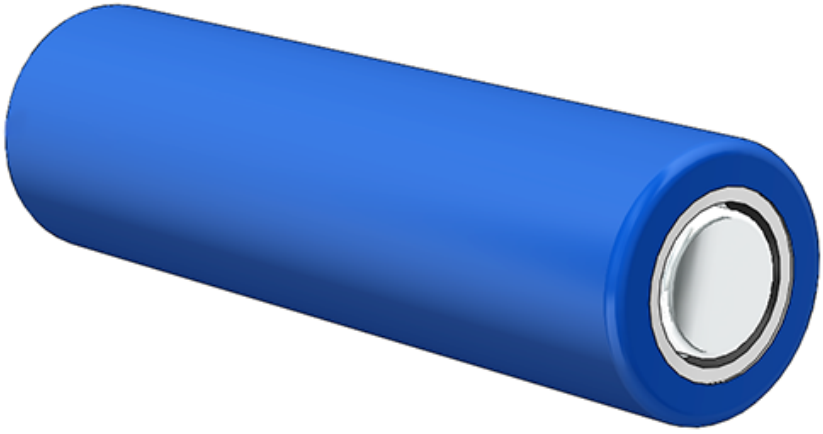
- 3.7V
- 18650
- 可充电的
- 锂离子电池
- 尖头电池
- 无保护板

注解:

- Robot HAT 无法给电池充电，需要购买电池充电器。
- 当 Robot HAT 上的两个电量指示灯熄灭时，表示电量过低，需要给电池充电。

尖头 vs 平头？

请选择尖头电池，以确保电池与电池座之间的连接良好。

尖头	平头
	

没有保护板？

建议使用没有保护板的 18650 电池。否则可能会因为保护板的过流保护而导致机器人断电停止运行。

电池容量？

为了让机器人长时间工作，尽量使用大容量电池。建议购买容量为 3000mAh 及以上的电池。

CHAPTER 6

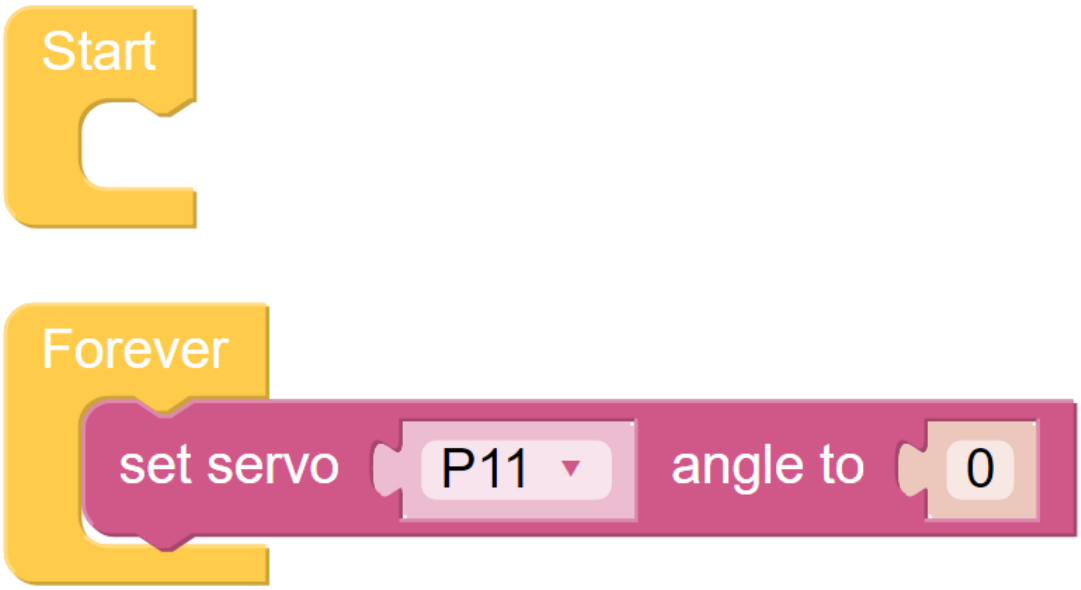
疑难解答

6.1 当使用 VNC 时，我被提示桌面暂时无法显示？

在终端，输入 `sudo raspi-config` 来改变分辨率。

6.2 安装 EzBlock 操作系统后，舵机不能转到 0°？

- 检查伺服电缆是否正确连接，Robot HAT 的电源是否打开。
- 按下复位按钮。
- 如果您已经在 EzBlock Studio 中运行了程序，那么 P11 的自定义程序就不再可用。您可以参考下图，在 EzBlock Studio 中手动编写程序，将伺服角度设置为 0。



6.3 如何在 EzBlock 重新校准机器人

在首页，点击左上角的产品连接图标。

进入到产品页面之后，点击 **设置**按钮。

进入设置页面后，点击 **校准**按钮就能进入到校准页面，然后按照提示来校准。

6.4 EzBlock 无法连接蓝牙？

- 先检查是否有给你的 SD 卡烧录*安装 EzBlock 镜像*。
- 打开产品的电源开关后，等蓝牙指示灯变得更亮并且出现”zi~” 声后，代表树莓派成功启动，此时再去连接。
- 检查您的移动设备的蓝牙是否打开。
- 检查 EzBlock 是否被允许访问设备的位置。
- 有些移动设备还需要打开位置服务。
- 检查电池电量。如果两个电源指示灯都关闭，或者只有一个指示灯在闪烁；电源电量低，请给电池充电。

- 如果以上方法都试过了，请尝试按下 RST 按钮，或重新启动产品和 APP。

6.5 APP 搜索到蓝牙，但无法连接

- 先检查是否有给你的 SD 卡烧录*安装 EzBlock 镜像*。
- 打开产品的电源开关后，等蓝牙指示灯变得更亮并且出现”zi~” 声后，代表树莓派成功启动，此时再去连接。
- 检查 ROBOT HAT 上的 BLE 或 USR 灯是否一直亮着（这意味着产品被其他设备连接），如果是，请断开其他设备的连接或重新启动产品。
- 如果以上方法都试过了，请尝试按下 RST 按钮，或者重新启动产品和 APP。

6.6 配置 WIFI 后 APP 无法连接

- 检查国家、账号和密码是否正确。
- 检查该 WIFI 的网络状态。
- 检查电源电量。如果两个电源指示灯都熄灭或只有一个电源指示灯闪烁，则说明电源电量不足，请给电池充电。
- 检查配置的 WiFi 和移动设备连接的 WiFi 是否相同。

6.7 为什么伺服机有时会无缘无故地返回到中间位置？

当舵机被结构物或其他物体挡住，无法到达预定位置时，舵机会进入断电保护模式，以防止舵机被过大的电流烧坏。

断电一段时间后，如果没有给伺服机提供 PWM 信号，伺服机将自动恢复到原来的位置。

CHAPTER 7

版权声明