

第 1 课 Hello, 行空板

行空板是一款由 DFRobot 专为 Python 编程教育而设计的开源硬件，它性能强大，操作简单。它自带 Python 环境和预装的 Python 库，让广大师生只需两步就能玩转 Python 编程。

下面，让我们来开启行空板之旅的第一节课吧！在屏幕上显示一些视觉互动效果。



任务目标

在行空板屏幕上显示文字和表情。



知识点

1. 认识行空板。
2. 理解行空板屏幕坐标知识。
3. 学习使用行空板图形化库书写文字、添加表情。

材料清单

硬件清单：



行空板 x1



Type-C USB 线 x1

软件使用：Mind+编程软件 x1

下载地址：<https://mindplus.cc/>

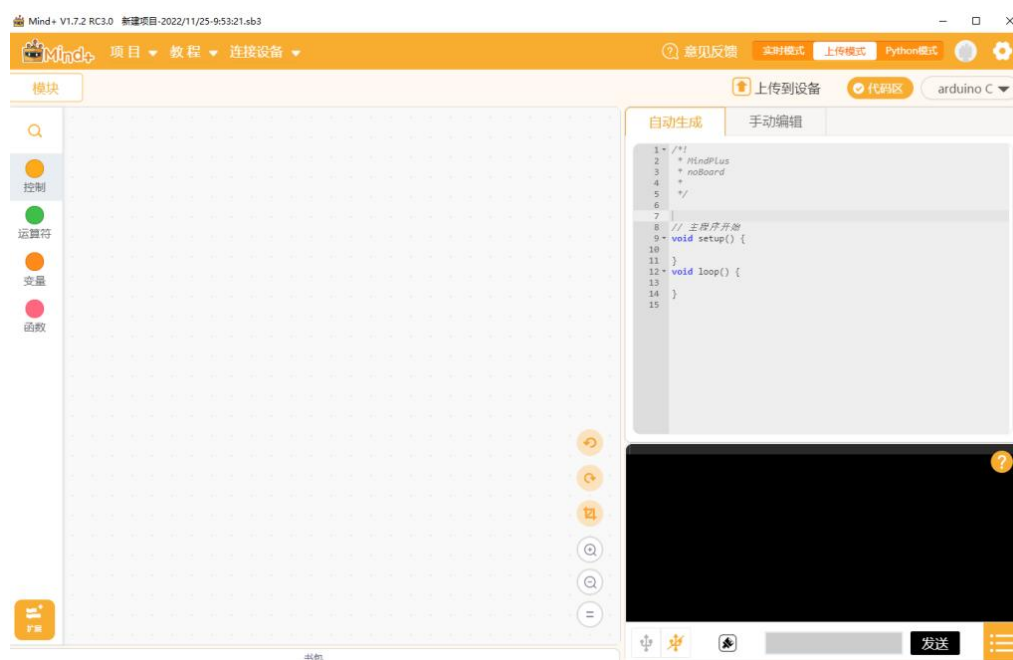


mind+软件基础使用

想要在行空板屏幕上显示文字和表情，我们就需要使用编程软件编写程序。在整套课程里我们使用的是 Mind+编程软件。它是一款集成了大量开源硬件的编程软件，它拥有非常多的图形化基础指令和扩展库，可以让大家轻松体验创造的乐趣。下面我们就基于 Mind+来学习基础的软件操作，为接下来的项目任务做好准备工作。

1. 认识 Mind+界面

1) 打开 Mind+，双击启动 mind+软件 (📄)，看到如下画面：



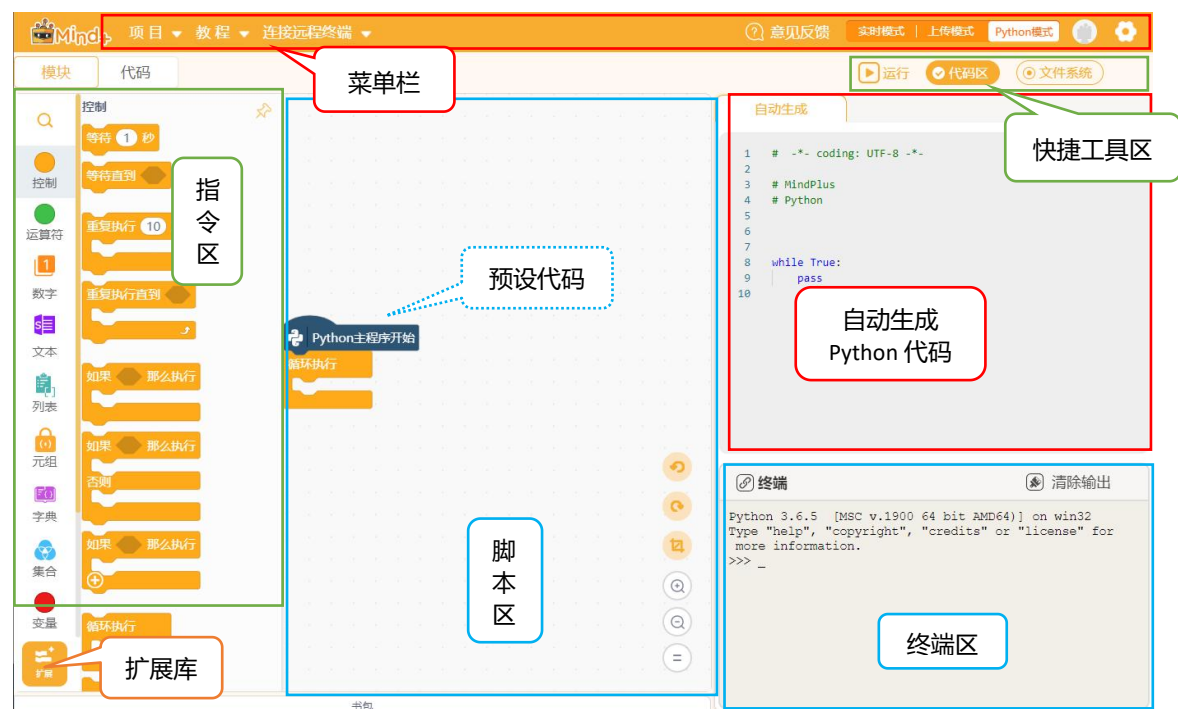
2) 点击右上角的“python 模式”。



3) 接下来点击左上角选择编程方式，“模块”表示图形化编程，“代码”表示 python 代码编程，这里选择“模块”，使用图形化指令完成我们的项目。



4) 完成上面的操作之后会看见编程操作界面，如下图：



注意：关于 Mind+界面中每个区域的详细介绍，见行空板图形化 python 入门教程“第一课 你好行空板”知识园地里的“Mind+相关界面介绍”。

2. 导入行空板库

点击“扩展库”，找到“官方库”下的“行空板”模块点击完成添加，点击返回后，就可以在“指令区”找到行空板，完成行空板加载。



提示：行空板加载完成后，原来的菜单栏会多出一个用来连接行空板的选项——“连接远程终端”。如下图。



3. 连接行空板

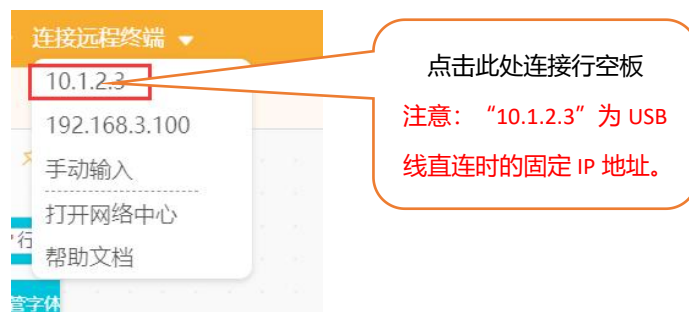
- 1) 使用 USB 线将行空板连接到计算机。



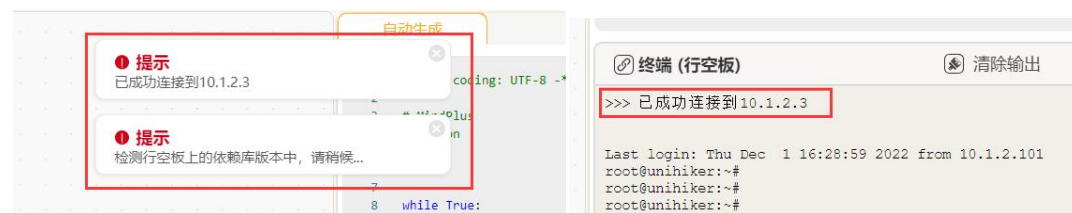
- 2) 确认行空板处于连接且开机状态，开始远程连接行空板。



- 3) 点击连接远程终端后，点击 10.1.2.3 连接行空板。



连接成功后会弹出“已成功连接到 10.1.2.3”的提示，终端区域也会显示该信息。同时，检测行空板上的依赖库版本，等待检测完成即可。



注意：连接不成功可以参考行空板图形化 python 入门教程“第一课 你好行空板”中“连接行空板”里的解决办法。

- 4) 连接成功后，点击“运行”按钮（如下图），Mind+会将 Python 代码发送到行空板上运行，你就可以在行空板上看到代码的运行效果了。



4. 保存文件

- 1) 点击“菜单栏”里的“项目”，并在出现的下拉菜单中点击“保存项目”。



- 2) 在弹出的界面中选择保存位置，输入你自己的文件名，保存类型“Mind+(*.sb3)”，点击保存即可。



动手实践

接下来，通过三个简单的小任务来学习如何在行空板上显示文字和表情

任务一：显示“行空板”

首先我们将学习显示文字的方法，并在行空板屏幕上显示预设文字“行空板”。

任务二：显示一串彩色文字

通过修改文字坐标，找到文字与文字之间的最佳间距，从而在行空板屏幕上显示一串彩色文字。

任务三：在行空板上添加表情

尝试在行空板底部添加一个表情。

任务一：显示“行空板”

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。

5.行空板连接成功 显示行空板 IP

1.切换至 python 模式

2.切换到图形化编程模式

3.出现预设代码

4.行空板库加载成功

```
1 # -*- coding: UTF-8 -*-
2
3 # MindPlus
4 # Python
5
6
7 while True:
8     pass
9
10
```

终端 (行空板) 清除输出

```
>>> 已成功连接到10.1.2.3
Last login: Wed Feb 8 11:30:35 2023 from 10.1.2.101
root@unihiker:~#
root@unihiker:~#
root@unihiker:~# _
```

3. 编写程序

在行空板指令下找到“对象名 () 显示文字 (“行空板”) 在 x (0) y (0) 字号 (20) 颜色 蓝”的指令，然后将它拖拽到指令区，并放在预设代码 Python 主程序开始的下面，如下图。

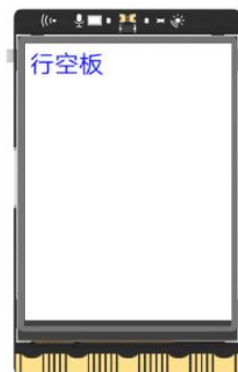


4. 参考程序



5. 程序运行

检查行空板是否已连接到 mind+, 检查无误后，点击右上角的运行按钮，观察行空板屏幕的显示效果，如下图。



6. 试一试

现在已经学会了在行空板上显示文字。那么请尝试修改上面的程序，在行空板上显示“Hello，行空板”的文字。

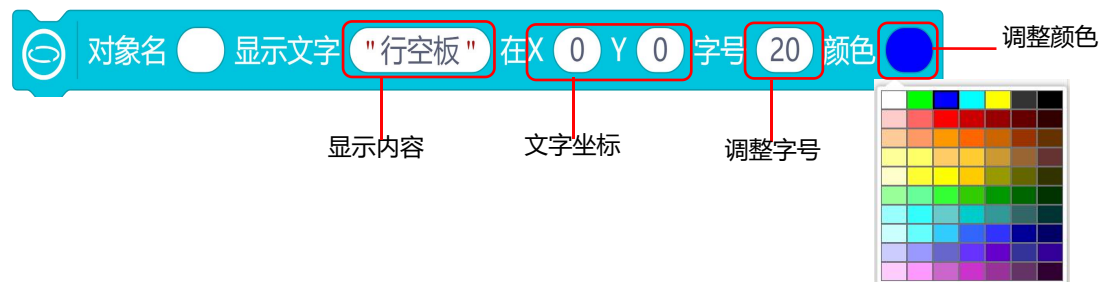
任务二：显示一串彩色文字

通过上一个任务，已经知道如何在坐标 (0, 0) 的位置显示“行空板”这三个字。接下来我们来学习如何在行空板上显示一串彩色的文字。

提示：因为硬件连接、软件准备和指令学习和任务一 一样，在此就不重复介绍（以后的课程也如此，不会介绍重复的部分）。

1. 编写程序

再次认识一下行空板上显示文字的指令。如下图。



(1) 确定文字颜色数量

因为最终要显示的文字是“Hello, 行空板!”和Hello, UNIHIKER! 每个汉字、单词和标点符号都有不同的颜色, 所以不能只使用一个指令。在本项目中一共有7个颜色, 就需要7个指令, 可以右键点击指令, 选择复制(如下图), 完成多个指令的搭建。



(2) 调试字之间的位置

每个汉字或符号有不同的位置, 在调试字之间的位置时, 你可以不必频繁观察行空板屏幕显示情况, 直接使用断开连接后的 Python 运行窗口快速进行坐标调试。操作步骤见下图。



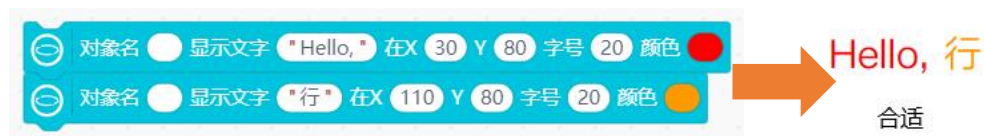
注意: 关于弹窗的更多问题, 参考行空板图形化 python 入门教程“第一课 你好行空板”知识园地里的“电脑端运行 Python 程序”。

接下来我们就可以利用这个弹窗完成字与字的间距调试。多个汉字间的距离, 一般只需要确定横向或者纵向两个汉字或符号的坐标, 记住它们的坐标差就可以通过计算获得所有汉字或符号的位置。

汉字以“行空”字样显示为例, 设置字号都为20, “行”字的位置在 X:110 Y:80, “空”字在右侧, 保证 Y 坐标不变, X 坐标向右增加, 合适的坐标情况如下图。



因为一个单词的长度和一个汉字的长度不一样，所以单词和汉字之间的距离需要相差 80 以上，同样 Y 坐标保持不变，合适的坐标情况如下图。



注意：字与字之间的调试过程参考行空板图形化 python 入门教程“第一课 你好行空板”里的“任务三”。

2. 参考程序



3. 程序运行

检查行空板是否已连接到 mind+，检查无误后，点击右上角的运行按钮，观察行空板屏幕的显示效果，如下图。



4. 试一试

现在已经学会了在行空板上显示彩色文字，那么可以试着将文字颜色修改成其他的颜色，例如深蓝，淡紫色。

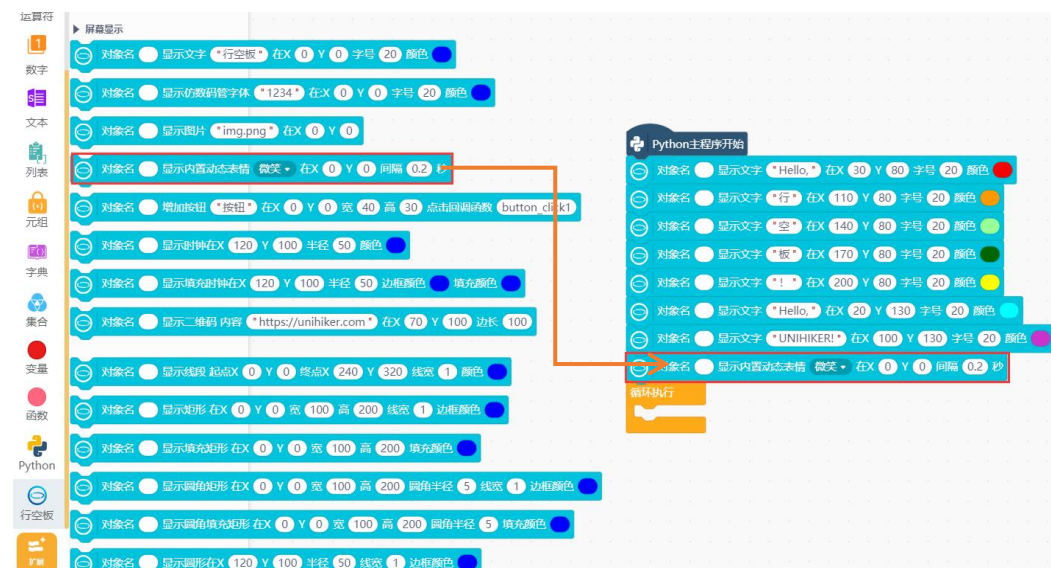
任务三：在行空板上添加表情

通过任务二，我们已经学会了如何在屏幕上显示一串彩色的文字，接下来我们来学习如何在行空板上显示内置表情。

1. 编写程序

(1) 显示动态表情

在行空板指令下找到“对象名 () 显示内置动态表情 (微笑) 在 X (0) Y (0) 间隔 (0.2) 秒”的指令，然后将它拖拽到任务二程序中，如下图的位置。



(2) 调试表情的位置

使用断开连接后的 Python 运行弹窗进行调试，调试方法和任务二一样。



2. 参考程序

Python主程序开始

对象名 显示文字 "Hello," 在X 30 Y 80 字号 20 颜色

对象名 显示文字 "行" 在X 110 Y 80 字号 20 颜色

对象名 显示文字 "空" 在X 140 Y 80 字号 20 颜色

对象名 显示文字 "板" 在X 170 Y 80 字号 20 颜色

对象名 显示文字 "!" 在X 200 Y 80 字号 20 颜色

对象名 显示文字 "Hello," 在X 20 Y 130 字号 20 颜色

对象名 显示文字 "UNIHAKER!" 在X 100 Y 130 字号 20 颜色

对象名 显示内置动态表情 眨眼 在X 0 Y 80 间隔 0.2 秒

循环执行

3. 程序运行

检查行空板是否已连接到 mind+, 检查无误后, 点击右上角的运行按钮, 观察行空板屏幕的显示效果, 如下图。



4. 试一试

现在我们已经学会在屏幕上显示“眨眼”的表情, 那么接下来就尝试用其他表情替换“眨眼”的表情。

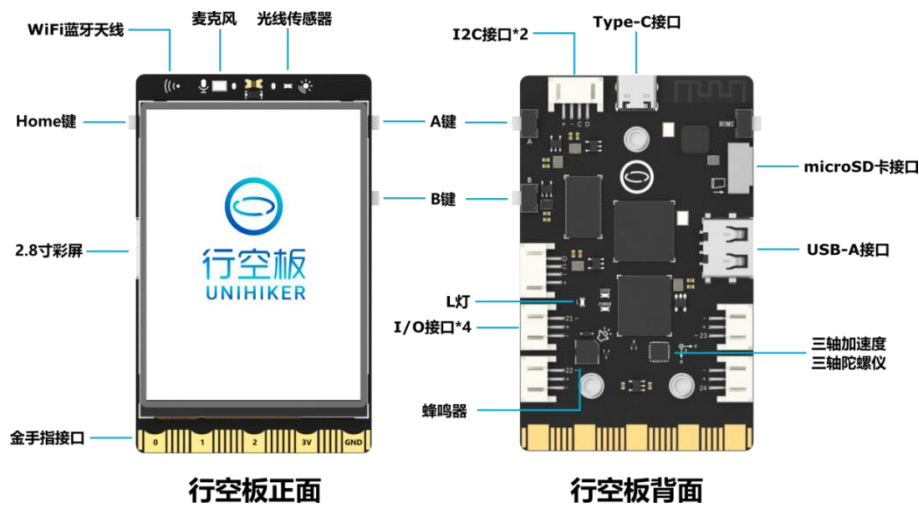
愤怒	紧张	平静	惊讶

知识园地

1. 行空板是什么?

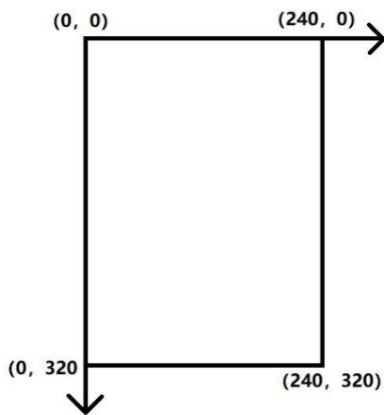
行空板又称 UNIHAKER, 它是一款国产教学用开源硬件, 它自带 Linux 操作系统和 Python 环境, 还预装了常用的 Python 库, 能够轻松胜任各种编程相关的开发场景, 如搭建物联网系统、体验人工智能应用、编写电子游戏、进行科学实验、设计声光互动、开发可穿戴设备等。

同时，行空板作为智能终端设备，其板子本身采用微型计算机架构，集成 LCD 彩屏、WiFi 蓝牙、多种常用传感器和丰富的拓展接口，如下图所示。



2. 行空板屏幕坐标

行空板屏幕是一块可触控的 LCD 彩屏，它的分辨率为 240×320 ，坐标原点为屏幕左上角，向右为 x 轴正方向，向下为 y 轴正方向。为了便于使图形等屏幕元素能够显示在指定位置，编程时我们可以通过设定 x 和 y 的坐标来实现。



3. 指令学习

在进行动手实践里，主要使用了以下的图形化指令。

Block	功能说明
	Python 图形化运行启动指令，必须要有此指令，才能控制代码运行。
	该指令主要用于行空板屏幕上显示文字，指令中可以设置文字的内容、显示位置、字号大小和文字颜色。
	循环执行指令，表示该积木块里的语句会一直执行。

选择内置表情

表情显示坐标

间隔时间

对象名

显示内置动态表情

微笑

主X 0 Y 0

间隔 0.2 秒

愤怒

紧张

平静

惊讶

睡觉

微笑

冒汗

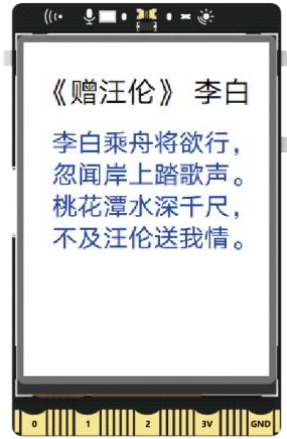
思考

眨眼

该指令主要用于行空板屏幕上显示内置的动态表情，指令中可以设置不同的表情、显示位置和图片切换时的等待间隔（动态表情播放速度）。

挑战自我

通过该课的学习后，你还有什么话想向行空板说呢？可以尝试着将它添加到屏幕上。例如显示一首唐诗。



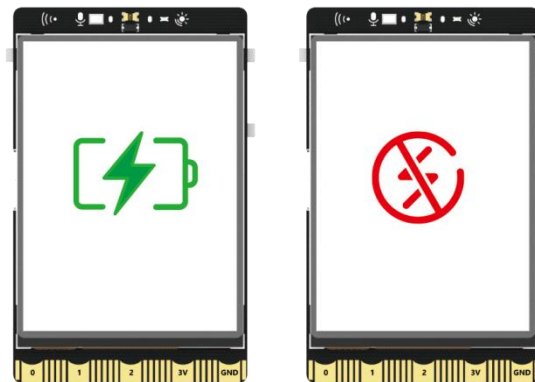
第 2 课 导电检测仪

在日常生活中，你会发现将身边的某些物体连接到电路中时，情况是不一样的。有些物体易导电，这样的物体叫做导体；有些物体不易导电，这样的物体叫绝缘体。本节课我们就来制作一个导电检测仪，用它来判断物体的导电性。



任务目标

用电导开关检测物体的导电性，并在行空板屏幕上显示导电结果。

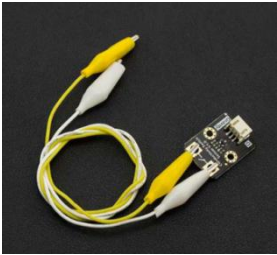


知识点

1. 掌握电导开关传感器的工作原理和使用方法。
2. 认识数字信号。
3. 认识设备里的输入元素。
4. 理解程序中对象的概念和命名规则。
5. 掌握行空板屏幕显示图片的方法。

材料清单

硬件清单:

	
行空板 x1	电导开关模块 x1
	
白色硅胶绞线 x1	Type- C USB 线 x1

软件使用: Mind+编程软件 x1

下载地址: <https://mindplus.cc/>

动手实践

导电检测仪，主要是利用电导开关检测物体是否导电，如果物体导电，行空板的屏幕上则显示导电的图片，反之，则显示不导电的图片。所以接下来，通过三个简单的小任务来学习如何制作一个导电检测仪。

任务一：布置初始画面

在此项任务中，我们通过学习行空板图片显示方法，根据“不导电”图片出现的位置，完成初始画面的显示。

任务二：切换图片

在完成初始化画面的显示后，我们将通过更新图片源的方法，来实现图片的切换效果。

任务三：使用电导开关检测物体的导电性

在这个任务中，我们会在行空板的 P24 脚上外接电导开关，用电导开关检测物体的导电性，并在行空板屏幕上显示导电结果。

任务一：布置初始画面

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。



3. 编写程序

因为行空板上电后，需要在屏幕上显示如下这张图。所以，接下来开始学习如何把它们显示到行空板屏幕上。



首先，背景图片在行空板屏幕上的显示需要 4 个步骤：找到图片、将图片加载进项目、使用指令显示图片、调整图片的位置。

(1) 找到图片，在附件资料中的“第 2 课 图片素材”文件夹中找到“不导电.png”图片文件；



- (2) 将图片加载进项目，在 Mind+ 的“快捷工具区”点击“文件系统”打开文件目录，将“不导电.png”图片拖入“项目中的文件”里。



拖入后，就已经成功将图片加入项目了，文件目录中会显示图片文件的完整名称（如下图）：

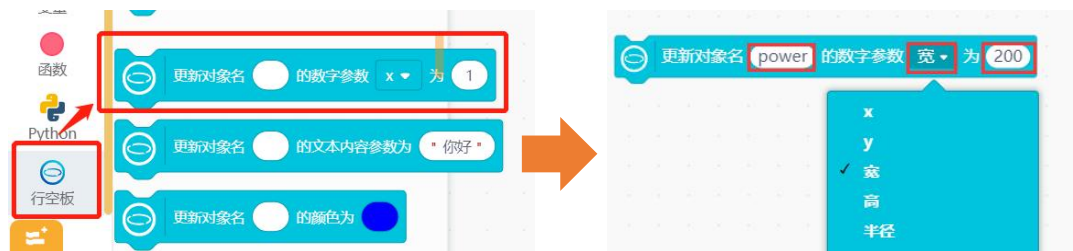


- (3) 使用指令显示图片，在指令区“行空板”分类下的“屏幕显示”里寻找含有“图片”关键字的指令，拖出对应显示图片的指令“对象名 () 显示图片 (img.png) 在 X (0) Y (0)”，然后修改“显示图片”为图片名称“不导电.png”，将指令放在预设代码“Python 主程序开始”的下面。



- (4) 现在已经能将图片显示在行空板的屏幕上了，接下来我们可以结合行空板屏幕坐标来设置图片的位置。因为图片的原始尺寸比较大（597 x 367），为了得到更好的显示效果，我们需要修改图片的宽或高，此时我们要给图片设置对象名“power”，修改图片的宽需要使用到“更新对象名 () 的数字参数 (x)”

为 (1) ” 指令，你可以在“行空板”指令分类下，寻找含有“数字参数”关键词的指令，拖出并修改成宽为 200。

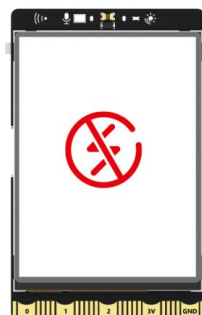


4. 参考程序



5. 程序运行

检查行空板是否已连接到 mind+，检查无误后，点击右上角的运行按钮，观察行空板屏幕的显示效果，如下图。



6. 试一试

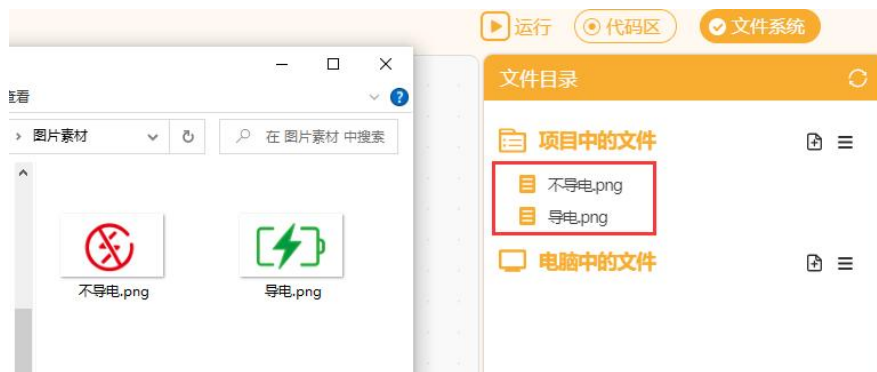
现在已经学会了在行空板上显示图片。那么请尝试修改上面的程序，在行空板上显示一个动态的效果图。

任务二：切换图片

通过上一个任务，已经完成了初始画面，接下来我们学习如何切换图片。

1. 编写程序

- (1) 因为在任务三里，我们需要根据检测的结果切换成对应的图片，所以需要将“第 2 课 图片素材”文件夹中的“导电.png”的图片加载进程序，然后使用指令在程序中更换图片。



- (2) 更换图片的指令在指令区“行空板”分类下的“屏幕显示”里，你可以寻找含有“更新图片”关键词的指令，拖出“更新图片对象（）图片源为（img2.png）”。



- (3) 使用此指令时，需要指明需要更换哪张图片，例如在任务一中，我们给图片设定的对象名为“power”，现在我们就要修改“power”这张图片的图片源了。



- (4) 与该项目相关的图片一共有 2 张，所以在最开始初始画面后我们需要 1 个更换图片指令。

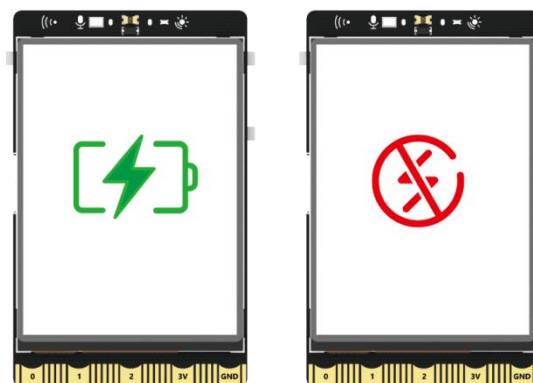


2. 参考程序



3. 程序运行

检查行空板是否已连接到 mind+, 检查无误后, 点击右上角的运行按钮, 观察行空板屏幕的显示效果, 如下图, 每隔 1 秒切换一张图片。



4. 试一试

现在已经学会了切换图片, 那么可以试着多用几张图片做一个连环画的效果。

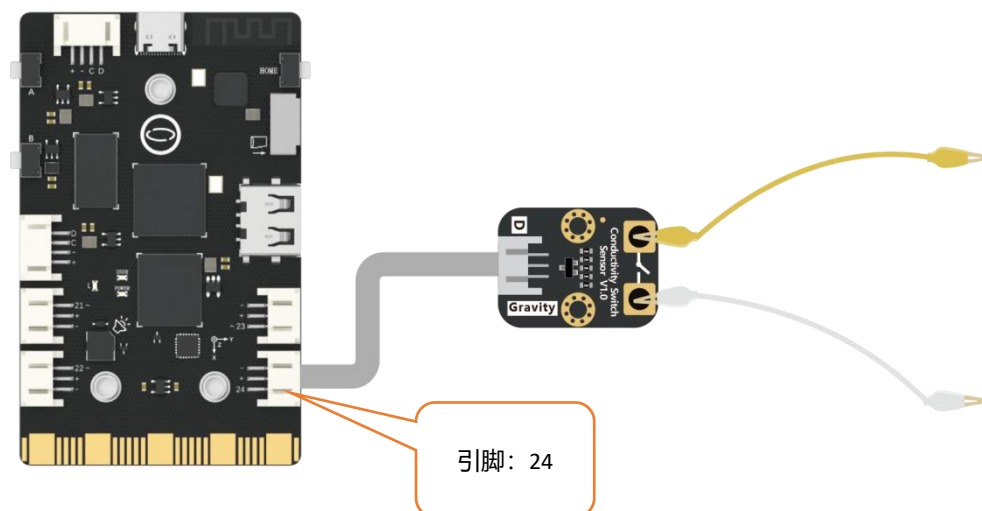
任务三：使用电导开关检测物体的导电性

通过任务二, 我们已经学会了如何切换图片, 接下来我们来学习使用电导开关检测物体的导电性。

1. 硬件连接

- (1) 使用连接线将电导开关连接到行空板上的 [P24 脚](#), 如下图。

注意：电导开关的相关介绍见“知识园地”。

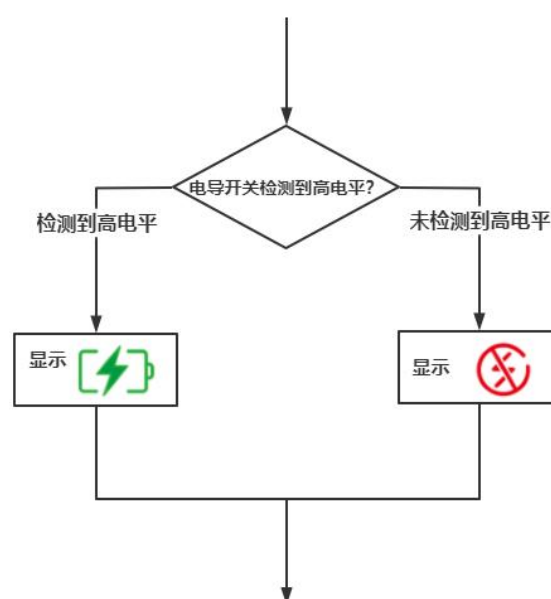


(2) 使用 USB 线将行空板连接到计算机。



2. 编写程序

在上一个任务中, 我们已经学会了如何更新图片源, 接下来我们需要利用电导开关检测物体的导电性。以圆木棍和铜线为例, 当检测的物体导电, 电导开关会发送一个**高电平信号**给行空板, 行空板屏幕会显示导电的图片; 如果检测的物体不导电, 电导开关会发送一个**低电平信号**给行空板, 行空板会显示不导电的图片。其程序执行流程图如下。

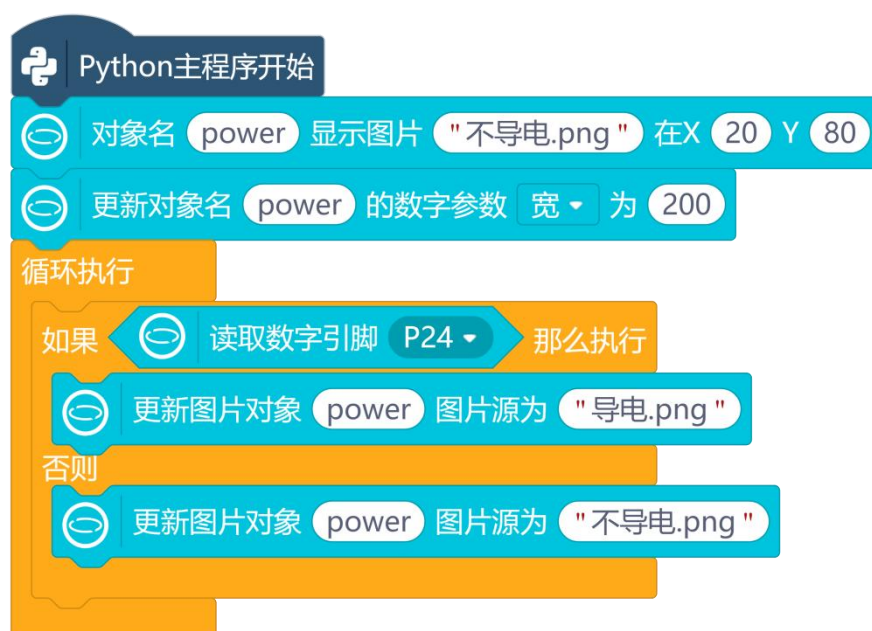


- (1) 要想实现上述功能，我们首先在控制指令区下找到“如果...那么执行...否则...”的指令，然后将它拖拽到任务一程序中，如下图的位置。



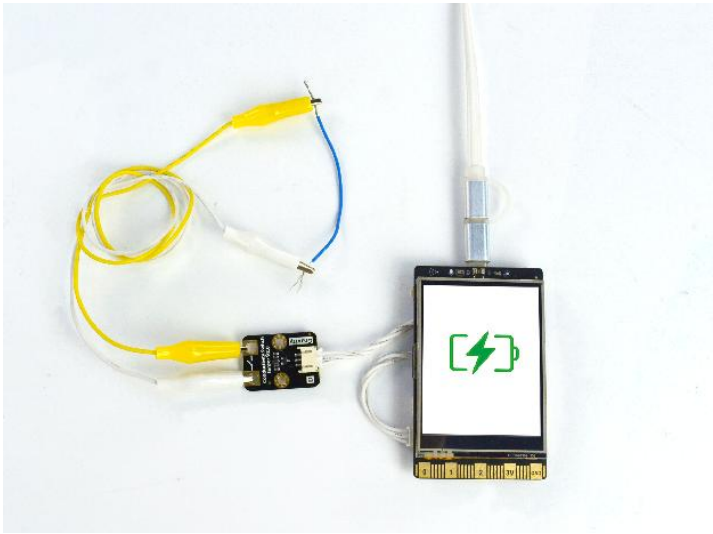
- (2) 然后设置判断条件；将“读取数字引脚（P24）”的指令拖拽到菱形的条件框中，最后再设置对应的显示结果，如参考程序。

3. 参考程序



4. 程序运行

检查行空板是否已连接到 mind+, 检查无误后，点击右上角的运行按钮，然后分别将导线和木棍夹在鳄鱼夹的中间后观察行空板屏幕的显示效果是否如下表。

物体	图片	导电性
导线		导电
木棍		不导电

5. 试一试

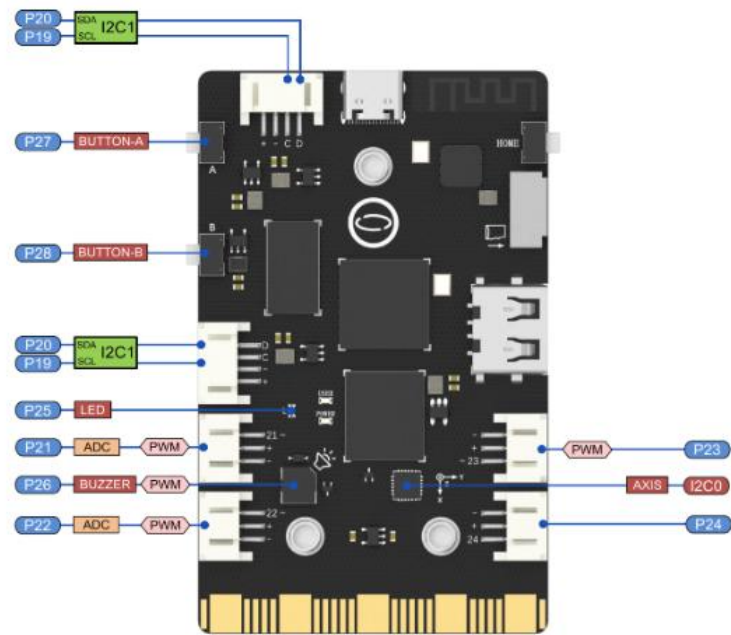
现在我们已经检测了导线和木棍的导电性，可以试着检测以下几个物体/物质的导电性？

物体/物质	纸巾	塑料管	钢尺	水	头发
导电性					

知识园地

1. 行空板引脚功能说明

行空板本身板载了 2 路 I2C 接口，4 个 3pin 接口，1 个 USB 接口和 1 个 SD 卡接口等。



接口/板载元件	对应引脚
4Pin I2C 口 * 2	I2C1 (独立 I2C 通道, 不与板载 I2C 元件共用)
3Pin I/O 口 * 4	P21、P22、P23、p24
LED 灯	P25
蜂鸣器	P26
按键 A	P27
按键 B	P28
三轴陀螺仪/加速仪	I2C0

注: 1.行空板正面的金手指引脚兼容 micro:bit。
2.其中 P21、P22 接口支持数字输入/输出、ADC 模拟输入、PWM 模拟输出; P23 接口支持数字输入/输出、PWM 模拟输出; P24 只支持数字输入/输出。
详细说明可查看官方文档: <https://wiki.unihiker.com/jianjie>

接口及板载元件在这里只做初步了解, 在后续使用过程中, 会有使用方法的具体讲解。本节课我们主要使用数字输入/输出接口。

2. 认识电导开关传感器

每当我们制作一个简单的装置时, 首先会考虑这三个元素: 输入、控制和输出。输入设备是用来收集信号, 控制器是用来接收并处理信号, 最后再由输出设备输出处理后的结果。以我们人为例, 五感就是输入信号, 当大脑接收到信号后再做出一个反应, 那输出的就是人的行为了。在本节课中, 电导开关传感器充当的角色就是**输入元素**。接下来我们就一起来认识一下这个传感器。

1) 电导开关工作原理

电导开关传感器主要用来检测物体是否具有导电性, 当电导开关的两个金属片之间的物体为导体, 则将整个电路连通, 传感器就会输出一个高电平 (**数字信号 1**)。如果金属片之间的物体为绝缘体, 则整个

电路断开，传感器输出的是一个低电平（数字信号 0）。

2) 电导开关使用方法

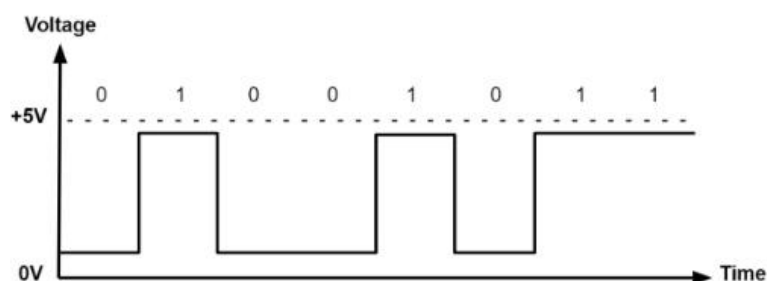
电导开关的使用方法见下图，将两根鳄鱼夹导线分别夹在物体的两端，另一端夹在传感器的两个圆孔中。



3. 什么是数字信号

前面我们了解到电导开关输出的是高低电平，表现出来的状态就是连通与断开，而这种状态在电子电路中属于数字信号。

数字信号是指用一组特殊的状态来描述信号，典型的就是当前用最为常见 0 和 1 两个二进制数字来表示的信号，之所以采用二进制数字表示信号，其根本原因是电路只能表示两种状态，即电路的通与断。在实际的数字信号传输中，当状态为 1（高电平）时，表示电路连通；状态为 0（低电平）时，则表示电路断开。



4. 理解程序里的“对象”

在今天的项目中，当检测的物体为绝缘体时，行空板上显示不导电的图片；当检测的物体为导体时，行空板上显示导电的图片。为了实现图片的自动切换，我们会在程序中建立“对象”，通过更新对象的图片源来实现图片的切换功能。接下来就来了解一下什么是程序中的“对象”。

在程序世界的“对象”是程序中具体的操作目标。比如，图片、文字等等这些具体的操作目标，都是对象。本节课，我们所介绍的对象是图片。



有了对象，给对象起好名字，就可以对对象的位置、颜色、文字内容、图片源等属性进行修改。修改的时候通常需要在建立好对象的基础上，搭配使用含有“更新对象”关键词的指令来完成操作，例如，要修改图片对象 power 的图片源为“导电.png”，需要如下两个指令即可完成：



注意：文字类对象的修改方法和更多对象的使用方法见行空板图形化 python 入门教程“第三课 西游记舞台剧”里的“程序中的对象”。

对象的命名规则

- 为了方便对对象进行操作，需要给对象命名，它的命名规则如下：
- a) 一般由数字、字母、下划线构成；
 - b) 不能由数字开头；
 - c) 命名不能使用 Python 的关键字，即 Python 中已经有特殊含义的词，如 True、False、def、if、elif、else、import 等；
 - d) 建议命名不要太长，尽量有意义，在使用 python 代码时，命名不要使用中文。

注意：使用图形化编程时，mind+会自动将中文命名的对象名转换成拼音。

5. 指令学习

在进行动手实践里主要使用了图片显示、更新对象数字参数以及更新图片对象等操作指令。

Block	功能说明
	该指令用于在行空板屏幕显示一个图片对象，在指令中可以设置对象名、图片文件名和图片位置。
	该指令用于改变指定对象的数字参数，在指令中需要写明改变的对象名、参数，以及修改参数值。
	该指令用于改变指定图片显示对象的图片内容，在指令中需要写明改变的对象名以及要修改的图片文件。
	该指令用于读取 P24 引脚上的数字信号（0 或者 1）。



Scratch 中的“如果...那么...否则...”条件语句块。该块包含三个部分：一个橙色的“如果”块，一个橙色的“那么”块，以及一个橙色的“否则”块。在“那么”块下方有一个输入框，标注为“语句1”。在“否则”块下方有一个输入框，标注为“语句2”。

该指令是一个条件语句，如果满足条件，则执行语句 1；如果不满足条件，则执行语句 2。

注意：条件语句的详细介绍及使用方法见行空板图形化 python 入门教程“第五课 密室逃脱游戏”里的“条件语句”。

挑战自我

你已经学会如何显示、切换图片以及使用电导开关，现在不妨使用电导开关完成一个“携手同心，共渡难关”的游戏。

携手同心游戏规则：所有参与的人手拉手一起越过面前的栏杆，同时保持栏杆不能掉。在此期间，如果参与人的手没松开，屏幕上则一直显示跳动的心。如果其中有人手松开了，表示闯关失败，行空板屏幕上会显示心碎的图片，并提示该队闯关失败。素材图片如下：

		
小爱心	大爱心	心碎

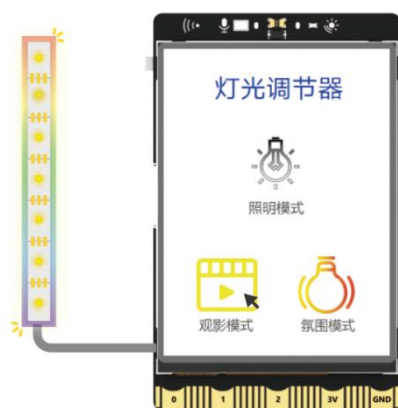
第 3 课 灯光调节器

在家里，无论您是在看电视、听音乐，或与家人在一起，或独自思考时，调节不同的灯光效果能为您创造舒适、宁静、和谐、温馨的气氛，能更深地体会生活，比如：柔和的光线能给您一个好心情，少而暗的光线能帮助您思考，多而亮的光线能使气氛更加热烈。而这些操作是非常方便的，你可以按住本地开关来调亮或调暗光线，也可以利用集中控制器或者是遥控器，只需要按下按键，就可以调节灯光的明暗亮度。接下来，我们就利用行空板和 RGB 灯带制作一款小型的灯光调节器。



任务目标

在行空板屏幕上设计几个按钮，当按下对应的按钮，RGB 灯带就显示对应的灯光效果。例如：照明模式下显示正常白光；观影模式下显示暗淡黄光，氛围模式下显示彩虹灯的效果。



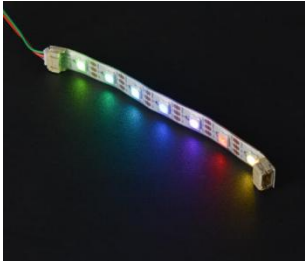
知识点

1. 掌握 RGB 灯带的使用方法及工作原理。
2. 掌握行空板基准点知识。
3. 学习使用回调函数。
4. 学习使用 for 循环。

5. 认识随机数。

材料清单

硬件清单：

	
行空板 x1	RGB 全彩灯带 x1
	
白色硅胶绞线 x1	Type-C USB 线 x1

软件使用：Mind+编程软件 x1

下载地址：<https://mindplus.cc/>

动手实践

灯光调节器，主要是在行空板屏幕上设计 3 个按钮，每个按钮代表一种灯光效果，比如照明模式的灯光效果就是简单的白光；观影模式就是暗淡的黄光；氛围模式则显示彩色的灯光。所以接下来，通过两个简单的小任务来学习如何制作一个灯光调节器。

任务一：设计灯光调节系统的控制界面

在此项任务中，利用图片素材完成灯光调节器的控制界面设计，然后将每个图片素材设置成按钮。

任务二：设计灯光效果

在任务一的基础上增加每种模式的灯光效果。

任务一：设计灯光调节系统的控制界面

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。



3. 编写程序

在程序开始之前, 先分析一下任务—需要实现什么功能? 首先在行空板上显示灯光调节器的控制界面, 然后通过回调函数, 将界面上的图片设置成按钮, 点击对应的按钮后, 在终端区显示对应的模式名称。灯光调节器的控制界面里用到的对象及对应的坐标如下图所示。



(1) 设计控制界面, 要设计灯光调节器的控制界面, 首先要将图片素材文件夹中的图片导入到文件系统中。

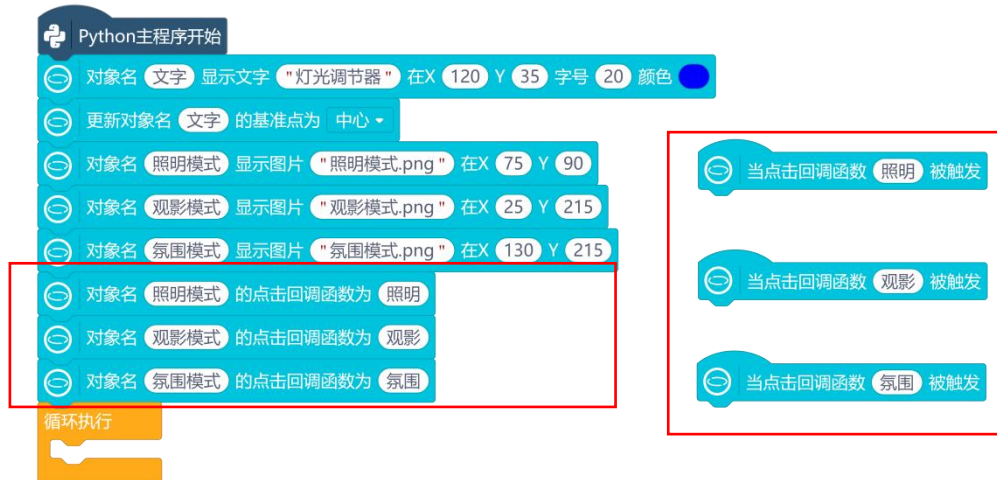


(2) 使用显示图片和显示文字的指令, 将“灯光调节器”这几个文字和文件系统里的图片显示到行空板上。为了让“灯光调节器”这几个文字居中, 我们需要使用“更新对象名 () 的基准点为 ()”的指令。然后根据上面分析的图片坐标设置对应的 XY 坐标即可。



注意：关于“行空板基准点”的相关介绍，见知识园地。

- (3) 使用“对象名 () 的点击回调函数为 (button_click1)”和“当点击回调函数 (button_click1) 被触发”指令为“照明模式”、“观影模式”、“氛围模式”对象增加点击回调函数，实现将图片设置成可点击的按钮。



注意：关于回调函数的相关介绍，见知识园地。

- (4) 接下来在“当点击回调函数 (button_click1) 被触发”指令下设置终端区输出内容，以此来验证回调函数是否设置成功。



4. 参考程序



5. 程序运行

检查行空板是否已连接到 mind+, 检查无误后, 点击右上角的运行按钮, 程序运行成功后, 分别点击“照明模式”、“观影模式”、“氛围模式”这几个图标, 终端区会打印“照明”、“观影”、“氛围”等文字。效果如下图。



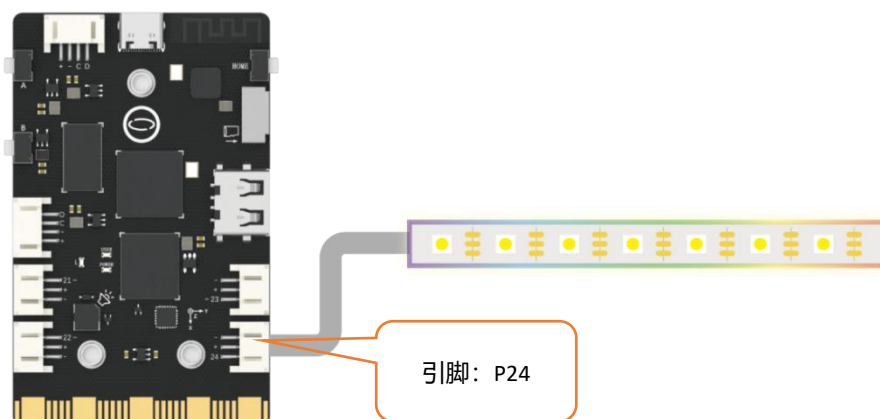
任务二：设计灯光效果

通过上一个任务, 已经设计好控制界面, 接下来我们将设计每种模式的灯光效果。

1. 硬件连接

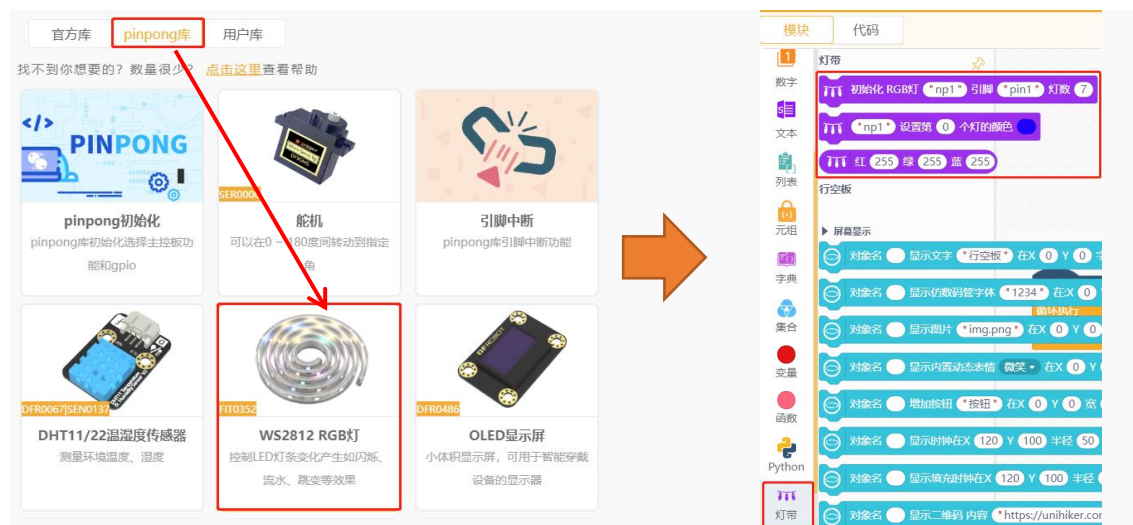
使用连接线将 WS2812 RGB 灯带连接到行空板上的 P24 脚, 如下图。

注意: 关于“WS2812 RGB 灯带”的相关介绍, 见[知识园地](#)。



2. 软件准备

- (1) 打开 Mind+, 按照任务一完成软件准备过程。
- (2) 添加 WS2812 RGB 灯库, 点击“扩展”, 在 pinpong 库中找到“WS2812 RGB 灯”并点击, 完成添加。



3. 编写程序

制作照明模式与观影模式的灯光效果

- (1) 要使用 RGB 灯带, 首先需要在 Python 主程序开始下面使用“初始化 RGB 灯 (np1) 引脚 (pin1) 灯数 (7)”指令, 对 RGB 灯带的引脚, 以及灯带的灯数进行初始化设置。



- (2) 在硬件连接图中, 灯带是连接到行空板的 P24 引脚, 下面就使用行空板引脚操作中的“行空板引脚 (P21(~A))”指令, 选择 P24 引脚, 然后将该指令放入“初始化 RGB 灯 (np1) 引脚 (pin1) 灯数 (7)”指令的引脚中, 最后设置灯带名称为“灯带”, 灯数为 7。



- (3) 设计照明模式灯光效果，在任务一中，当“照明”的回调函数被点击后，终端区会显示“照明”的文字，如果将显示文字的的代码替换成 RGB 灯带的颜色显示白色，就可以实现当“照明模式”被点击后，灯带就显示白光。程序设置如下图。



为了实现每一颗灯珠全亮，首先建立一个变量“number”，然后使用 for 循环指令，设置“灯珠个数”范围为 0-6，步长为 1 来控制每个灯珠都显示白色。

注意：

- 1.在 python 中，变量命名一般遵守蛇形命名法，例如：`book_id`，`book_store_count`。
- 2.关于“变量”的详细介绍，见行空板图形化 python 入门教程“第五课 密室逃脱游戏”的知识园地。）

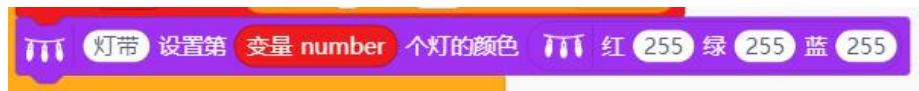


- (4) 设计观影模式灯光效果，在“观影”的回调函数下设置灯带的显示颜色为黄色的同时并设置灯带亮度为 50。



制作氛围模式灯光效果

- (1) 氛围灯模式下的灯光是随机变换的彩色，所以我们需要使用“红 (255) 绿 (255) 蓝 (255)”指令，该指令使用的是 RGB 三原色调色的原理，将这三种色光以不同比例混合，就能得到其他更多的颜色。如果想让灯带的颜色显示成彩色，需要让红绿蓝指令与“(np1) 设置第 (0) 个灯的颜色 蓝色”指令搭配使用，如下图。



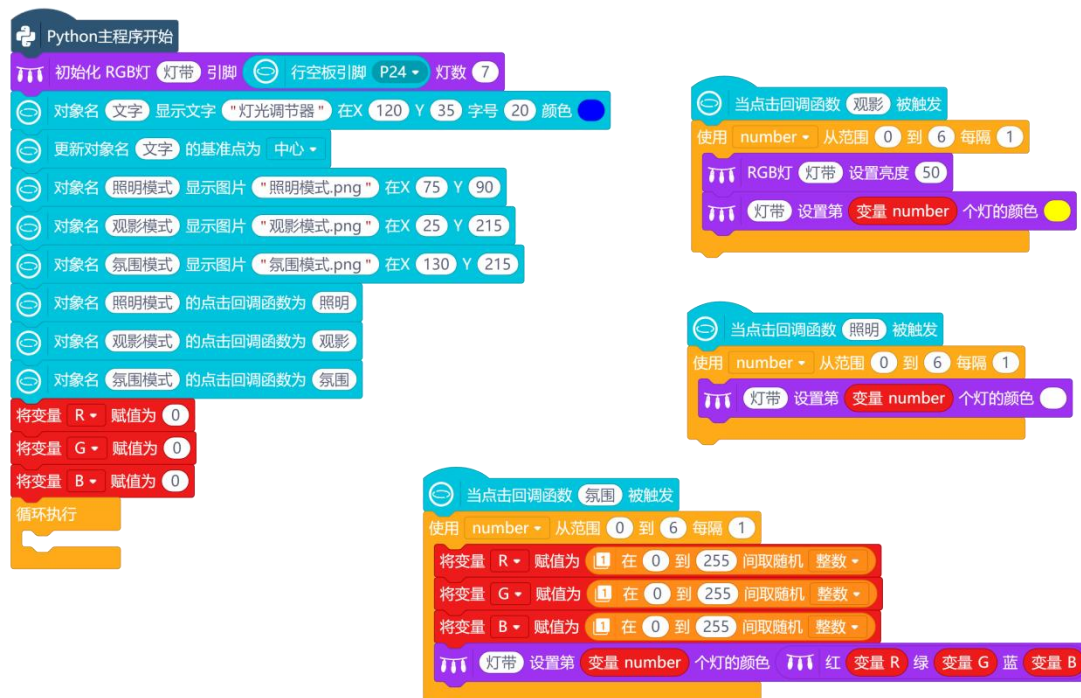
- (2) 为了实现随机彩色的效果，就需要随机改变红绿蓝的数值来设置灯带的颜色。首先新建三个变量“变量 R”、“变量 G”、“变量 B”并在 Python 主程序开始下面，将这三个变量的值设置为 0。然后当“氛围”的回调函数被触发后，使用数字类型中的“在 (1) 到 (10) 间取随机 整数”指令让这三个变量，分别生成一个 0-255 的随机整数。（注意：关于“随机数”的相关介绍，见知识园地。）



- (3) 将“变量 R”、“变量 G”、“变量 B”，分别放入“红 (255) 绿 (255) 蓝 (255)”指令中，氛围模式灯光效果程序如下：



4. 参考程序



5. 程序运行

点击右上角的运行按钮，观察行空板屏幕的显示效果，点击界面上的照明模式，RGB 灯带显示白色的光；点击界面上的观影模式，RGB 灯带显示暗淡的黄光；点击氛围模式，RGB 灯带显示彩色的光。



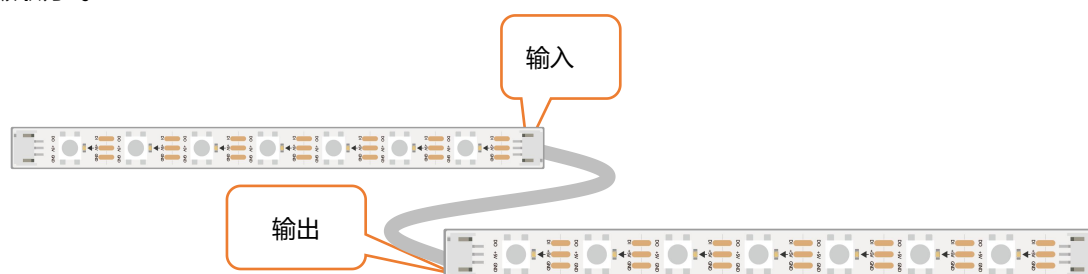
知识园地

1. 认识 RGB 灯带

在第二课的时候，我们认识了什么是**输入元素**，这节课就来认识一下**输出元素**——**WS2812 RGB 全彩灯带**，接下来简称 RGB 灯带。RGB 灯带是由 7 颗全彩 LED 组成，灯带两头留有 PH2.0-3P 接口，一端为输入，一端为输出。输出口用于灯带之间的级联。如何级联呢？使用白色硅胶线将灯带的输出端与另一个灯带的输入端连接起来。



级联方式:



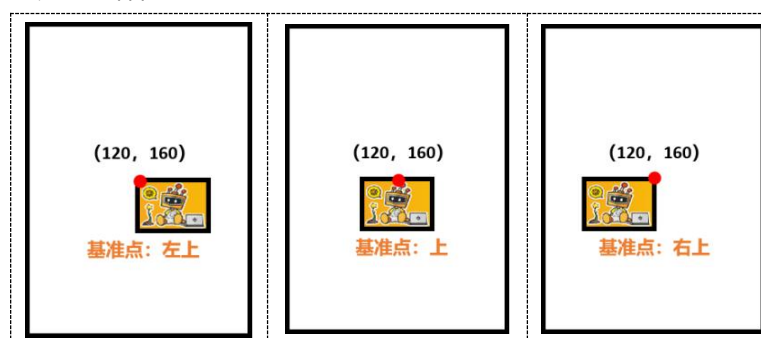
其实 RGB 灯是属于 LED 灯的一种，RGB 灯内部集成了红、绿、蓝这三个 LED 灯。可以通过调节这三个 LED 中的每个灯的亮度就能产生不同的颜色，这样就可以让 RGB 灯带显示各种颜色了。如何实现显示不一样的颜色呢？在 mind+ 中，我们可以利用 “(np1) 设置第 (0) 个灯的颜色 蓝色” 指令选择颜色。

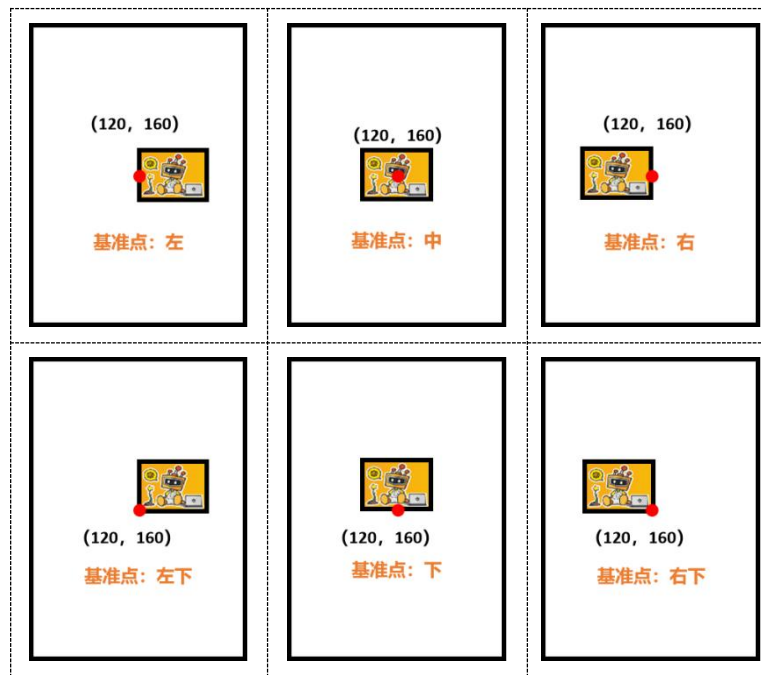
设置 RGB 灯颜色指令
是将常用颜色的 RGB
值封装成对应的颜色
供我们选择。

除以上使用方法外，我们还可以使用 “红 (255) 绿 (255) 蓝 (255)” 指令设置红、绿、蓝 0~255 之间的 256 级亮度显示，完成 16777216 (256*256*256) 种颜色的全真色彩显示。

2. 行空板基准点

行空板一共设定了 9 个基准点，即上、下、左、右、中、左上、左下、右上、右下。也就是说，如果要在行空板屏幕的中间位置显示一张图片，即下图中的红点位置显示，坐标为 (120, 160)，根据基准点的不同，会出现以下 9 种位置情况。





3. 认识回调函数

回调函数就是一个被作为参数传递的函数，简单来说就是把一段可执行的代码（一个函数），当成参数传给另一个函数，这个行为就是回调函数。它和一般函数的区别就在于，一般函数接收的参数是一般类型；回调函数接收的参数是一个函数。例如：一般函数：`function a(int a, String b)`；特殊函数：`function b(function c)`，函数 `c` 就叫回调函数。

注意：回调函数的执行流程见行空板图形化 python 入门教程“第四课 情绪卡片”里的“回调函数”。

4. 学习使用 for 循环

在项目程序中，我们使用了 for 循环指令，该指令调用的是 Python 内置的 `range()` 函数创建一个整数序列。指令中，需要设置三个参数，开始值、停止值、步长。这个指令可以实现循环从开始值计数到达停止值，一旦到达停止值就立刻退出循环。也就是说，这个循环指令是按照变量 `my variable` 的变化控制循环次数。



什么是 `range()` 函数？我们将这条指令结合 Python 代码来分析，其中 `my variable` 是变量，`range()` 是函数，`range` 表示在指定范围内，等间隔排列的整数集合对象。例如，现在设定指定开始值（1），停止值（9），以及步长（2），`range()` 函数就是将 1-9 的数字进行等间隔排列 {1、2、3、4、5、6、7、8、9}，步长就是控制变量 `my variable` 从 1 开始每次递增 2，从而实现将列表中编号为 1、3、5、7、9 这五个数据取出来。

注意：更多循环语句的介绍见行空板图形化 python 入门教程“第十二课 肺活量数据可视化报告”里的

“Python 中的循环语句”。

5. 认识随机数

什么是**随机数**？随机数可以简单理解为随机的数字，是从一组可能的值中提取出来的，并且每个可能的值被提取的概率是一样的。举个例子，很多人都玩过掷骰子的游戏，通过掷骰子的方式生成的点数，就是一个 1 到 6 之间的随机数。



例如下面程序，使用随机数指令，设置随机数的范围为 1-10。这个指令就如同一个有 10 个面的骰子，掷骰子生成的点数，就是这个指令生成的随机数。



6. 指令学习

动手实践里主要使用了以下操作指令。

Block	功能说明
	该指令用于选择行空板上的引脚对象。
	该指令用于获取范围内的任意整数或小数。
	该指令用于初始化灯带或灯带的引脚与灯珠数。
	该指令用于设置指定灯珠的颜色。
	该指令用于通过 RGB 三原色调色原理，设置灯珠显示颜色。
	该指令用于设置灯带或灯带的显示亮度。
	该指令用于给指定对象增加一个点击的回调函数，指令中需要写明对象名、回调函数名。
	该指令需要和设置回调函数指令搭配使用，并且函数名需要保持一致。

	该指令用于修改指定对象的基 准点。使用时，需写明要修改的对象名称。
	for 循环指令，使用内置的 range 函数创建一个整数序列，然后通过 for 循环遍历。
	初始化变量的值
	变量 R

挑战自我

现在我们已经学会了如何通过行空板点亮灯带了。接下来，请大家在任务二的程序基础上修改代码，让氛围模式下的灯光效果变成彩色的跑马灯。

提示：使用标志变量的方法，将跑马灯的程序放在重复执行指令中，并在点击回调函数触发指令下，定义变量 flag 为全局变量。



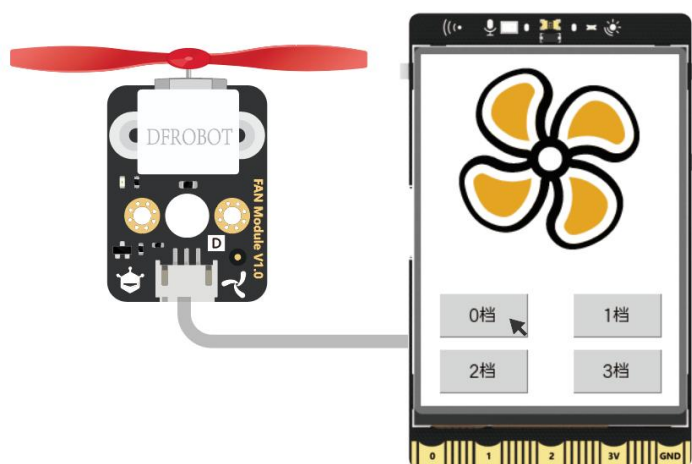
第 4 课 变速风扇

随着科技的发展，市面上降温的家电也越来越多了，比如：空调，风扇、空调扇等，虽然现在空调的制冷效果远超过风扇，但还是有很多人在夏天把电风扇当做避暑降温的常用家电。因为它不仅能清凉解暑还能让室内的空气流通。接下来我们就利用行空板和风扇模块，制作一款可根据不同档位改变风速的风扇。



任务目标

在行空板屏幕上设计 4 个档位按钮，当按下对应的档位，风扇就改变成不同的风速。例如：0 档表示风扇停止；1 档表示轻风；2 档表示微风；3 档表示强风。



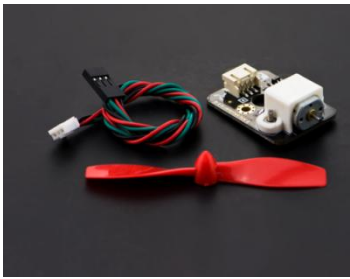
知识点

1. 掌握风扇模块的使用方法。
2. 认识 PWM 信号输出。
3. 了解带参函数。

4. 认识全局变量。

材料清单

硬件清单：

	
行空板 x1	风扇模块 x1
	
白色硅胶绞线 x1	Type-C USB 线 x1

软件使用：Mind+编程软件 x1

下载地址：<https://mindplus.cc/>

动手实践

变速风扇，主要是通过行空板屏幕上的 4 个按钮去控制风扇模块上的电机转动不同的速度。接下来，通过三个简单的小任务来学习如何制作一个变速风扇。

任务一：设计风扇的控制界面

在此项任务中，利用图片素材和回调函数设计变速风扇的控制界面。

任务二：设计风扇转动的动画效果

在完成控制界面的设计后，我们将通过切换不同的图片素材，来实现屏幕上风扇转动的动画效果。

任务三：使用 PWM 信号控制风扇模块的转速

这一步我们需要在行空板上外接风扇模块，并通过 PWM 信号控制它的转速，从而完成变速风扇这个项目。

任务一：设计风扇的控制界面

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。



3. 编写程序

首先观察一下，控制界面上都包含哪些元素？风扇、0~3 档的按钮，效果如下图。



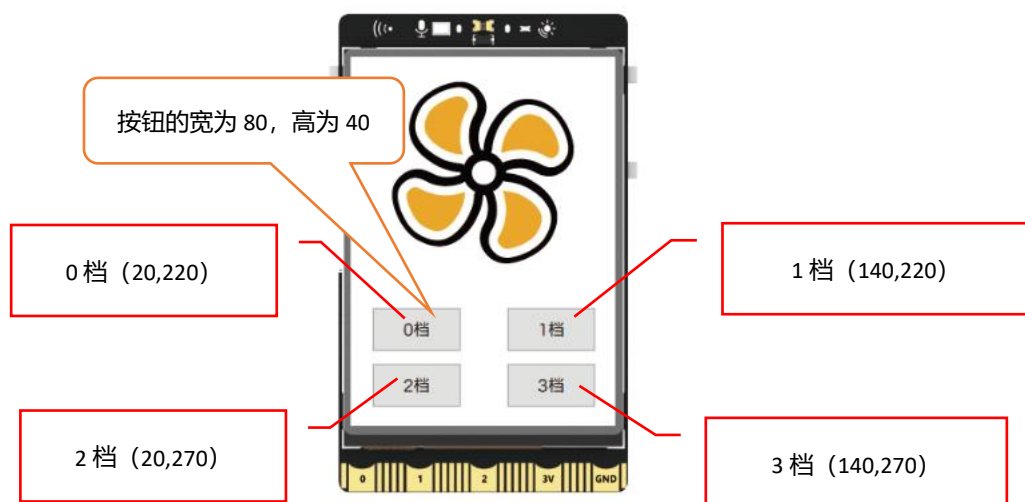
(1) 设计控制界面，首先要将图片素材文件夹中的“风扇 1.png”导入到文件系统中。



- (2) 使用显示图片指令，将文件系统里的图片“风扇 1.png”以中心为基准点显示到 X 120 Y 100 这个坐标上，如下图。

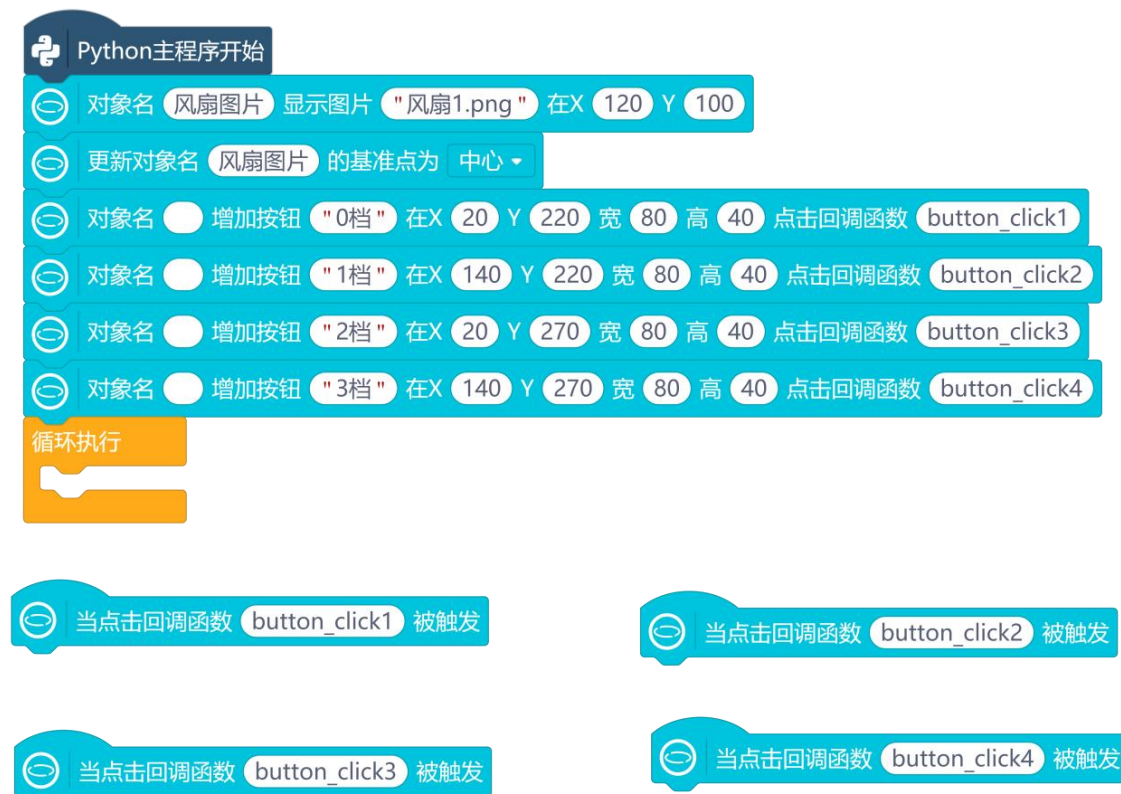


- (3) 使用“对象名 () 增加按钮 (‘按钮’) 在 x0y0 宽 () 高 () 点击回调函数 (button_click1)”和“当点击回调函数 (button_click1) 被触发”指令添加“0 档、1 档、2 档、3 档”这几个屏幕按钮。每个按钮的 XY 坐标如下图。





4. 参考程序



5. 程序运行

检查行空板是否已连接到 mind+, 无误后, 点击右上角的运行按钮, 程序运行成功后, 行空板屏幕上会显示如下图的界面。



任务二：设计风扇转动的动画效果

不知道大家有没有听说过翻书动画？今天我们要实现的效果和它类似，都是通过切换多张连续动作的图片让图画里的人物或物体动起来。接下来我们将通过切换不同图片的方法来设计风扇转动的动画效果。

1. 准备工作

按照任务一的方法，使用 USB 线将行空板连接到计算机，并准备好软件。

2. 编写程序

在程序开始之前，先分析一下每个档位需要什么样的动画？通过观察家用风扇的工作情况，发现 0 档是关闭风扇，扇叶则不会转动；1 档表示轻风，扇叶转动得比较慢；2 档表示微风，扇叶转动得较快；3 档表示强风，扇叶转动得最快。接下来我们将一起来学习如何制作风扇不同转速的动画效果。

- (1) 如果想要风扇转动起来，就需要多张不同角度的图片，在任务一的基础上将图片素材文件夹中的“风扇 2.png，风扇 3.png”导入到文件系统中。



- (2) 图片导入成功后，接下来就是更新图片，首先在指令区找到更新图片对象 () 图片源为 ("img.png") 的指令，然后按照风扇 1—风扇 2—风扇 3 的顺序更新“风扇图片”的图片源。为了区分不同档位下风速的变化，我们可以在每个更新图片的指令面前放一个等待 () 秒的指令，然后根据风速的大小设

置时间，风速越大时间越小。

风扇关闭动画效果



0 档，关闭风扇，扇叶不转动，所以，直接显示“风扇 1”的图片。

轻风动画效果



1 档，轻风，扇叶转动得慢。

微风动画效果



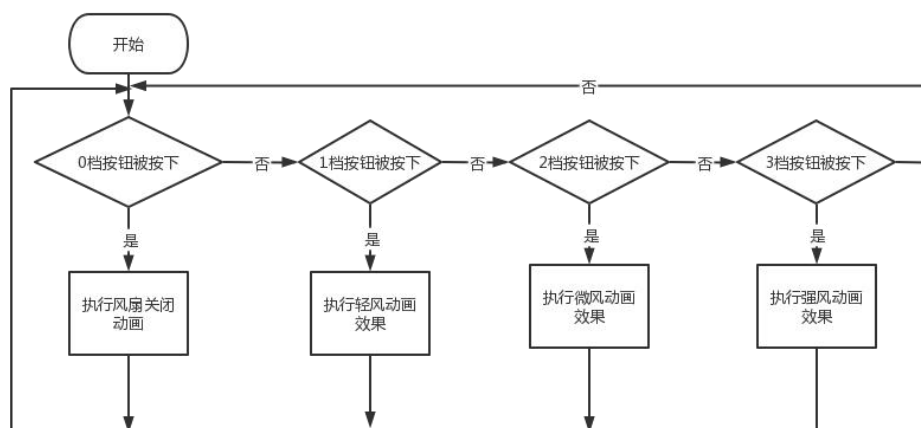
2 档，微风，扇叶转动得较快。

强风动画效果



3 档，强风，扇叶转动得最快。

(3) 通过前面的程序分析，每按下一个档位按钮，就循环执行对应的动画效果，直到按下其他的按钮才退出当前的动画效果，它们之间的执行流程如下图。



根据流程图描述的功能，我们将对应的动画程序放在“当点击回调函数（button_click1）被触发”的指令下，程序运行后你会发现所有的动画效果在按钮被点击后执行一次就不再执行了，而理想效果是按钮被点击后，动画效果会一直重复执行，直到其他按钮被点击后，才会退出当前的动画效果。

想让所有的动画效果重复执行，我们可以使用标志变量的方法，将每个动画效果的程序放在循环执行的指令中。首先新建一个变量 flag，并设置该变量的初始值为 0，然后在每个“当点击回调函数（）被触发”指令下，将变量 flag 定义为全局变量。当回调函数被触发后就设置变量 flag 的值为对应的档位数，例如 0 档：flag=0；1 档：flag=1。

注意：关于“全局变量”的介绍见本课知识园地。

Python主程序开始

对象名 风扇图片 显示图片 "风扇1.png" 在X 120 Y 100

更新对象名 风扇图片 的基准点为 中心

对象名 增加按钮 "0档" 在X 20 Y 220 宽 80 高 40 点击回调函数 button_click1

对象名 增加按钮 "1档" 在X 140 Y 220 宽 80 高 40 点击回调函数 button_click2

对象名 增加按钮 "2档" 在X 20 Y 270 宽 80 高 40 点击回调函数 button_click3

对象名 增加按钮 "3档" 在X 140 Y 270 宽 80 高 40 点击回调函数 button_click4

将变量 flag 赋值为 0

初始化标志变量 flag

循环执行

定义变量 flag 为全局变量

当点击回调函数 button_click1 被触发

全局global 变量 flag

将变量 flag 赋值为 0

点击 0 档按钮后，设置标志变量 flag=0

当点击回调函数 button_click2 被触发

全局global 变量 flag

将变量 flag 赋值为 1

点击 1 档按钮后，设置标志变量 flag=1

当点击回调函数 button_click3 被触发

全局global 变量 flag

将变量 flag 赋值为 2

点击 2 档按钮后，设置标志变量 flag=2

当点击回调函数 button_click4 被触发

全局global 变量 flag

将变量 flag 赋值为 3

点击 3 档按钮后，设置标志变量 flag=3

(4) 设置好全局变量后，在循环执行里面判断每个档位设置的变量值，如果判断的结果为真时，则执行对应的动画效果。例如：如果 flag=2，表示“2 档”的按钮被触发，则执行步骤 2 里的微风动画程序。

注意：该步骤会用到运算符，关于“运算符”的介绍见行空板图形化 python 入门教程的“第五课 密室逃脱游戏”的知识园地。

Python主程序开始

对象名 风扇图片 显示图片 "风扇1.png" 在X 120 Y 100

更新对象名 风扇图片 的基准点为 中心

对象名 增加按钮 "0档" 在X 20 Y 220 宽 80 高 40 点击回调函数 button_click1

对象名 增加按钮 "1档" 在X 140 Y 220 宽 80 高 40 点击回调函数 button_click2

对象名 增加按钮 "2档" 在X 20 Y 270 宽 80 高 40 点击回调函数 button_click3

对象名 增加按钮 "3档" 在X 140 Y 270 宽 80 高 40 点击回调函数 button_click4

将变量 flag 赋值为 0

循环执行

如果 变量 flag = 0 那么执行

更新图片对象 风扇图片 图片源为 "风扇1.png"

当标志变量 flag=0 之后, 才执行风扇关闭的动画效果。

当点击回调函数 button_click1 被触发

全局global 变量 flag

将变量 flag 赋值为 0

如果 变量 flag = 1 那么执行

等待 0.1 秒

更新图片对象 风扇图片 图片源为 "风扇1.png"

等待 0.1 秒

更新图片对象 风扇图片 图片源为 "风扇2.png"

等待 0.1 秒

更新图片对象 风扇图片 图片源为 "风扇3.png"

当标志变量 flag=1 之后, 才执行轻风动画效果。

当点击回调函数 button_click2 被触发

全局global 变量 flag

将变量 flag 赋值为 1

如果 变量 flag = 2 那么执行

等待 0.05 秒

更新图片对象 风扇图片 图片源为 "风扇1.png"

等待 0.05 秒

更新图片对象 风扇图片 图片源为 "风扇2.png"

等待 0.05 秒

更新图片对象 风扇图片 图片源为 "风扇3.png"

当标志变量 flag=2 之后, 才执行微风动画效果。

当点击回调函数 button_click3 被触发

全局global 变量 flag

将变量 flag 赋值为 2

如果 变量 flag = 3 那么执行

等待 0.01 秒

更新图片对象 风扇图片 图片源为 "风扇1.png"

等待 0.01 秒

更新图片对象 风扇图片 图片源为 "风扇2.png"

等待 0.01 秒

更新图片对象 风扇图片 图片源为 "风扇3.png"

当标志变量 flag=3 之后, 才执行强风动画效果。

当点击回调函数 button_click4 被触发

全局global 变量 flag

将变量 flag 赋值为 3

3. 参考程序

Python主程序开始

```

Python主程序开始
对象名 风扇图片 显示图片 "风扇1.png" 在X 120 Y 100
更新对象名 风扇图片 的基准点为 中心
对象名 增加按钮 "0档" 在X 20 Y 220 宽 80 高 40 点击回调函数 button_click1
对象名 增加按钮 "1档" 在X 140 Y 220 宽 80 高 40 点击回调函数 button_click2
对象名 增加按钮 "2档" 在X 20 Y 270 宽 80 高 40 点击回调函数 button_click3
对象名 增加按钮 "3档" 在X 140 Y 270 宽 80 高 40 点击回调函数 button_click4
将变量 flag 赋值为 0
循环执行
如果 变量 flag = 0 那么执行
    更新图片对象 风扇图片 图片源为 "风扇1.png"
如果 变量 flag = 1 那么执行
    等待 0.1 秒
    更新图片对象 风扇图片 图片源为 "风扇1.png"
    等待 0.1 秒
    更新图片对象 风扇图片 图片源为 "风扇2.png"
    等待 0.1 秒
    更新图片对象 风扇图片 图片源为 "风扇3.png"
如果 变量 flag = 2 那么执行
    等待 0.05 秒
    更新图片对象 风扇图片 图片源为 "风扇1.png"
    等待 0.05 秒
    更新图片对象 风扇图片 图片源为 "风扇2.png"
    等待 0.05 秒
    更新图片对象 风扇图片 图片源为 "风扇3.png"
如果 变量 flag = 3 那么执行
    等待 0.01 秒
    更新图片对象 风扇图片 图片源为 "风扇1.png"
    等待 0.01 秒
    更新图片对象 风扇图片 图片源为 "风扇2.png"
    等待 0.01 秒
    更新图片对象 风扇图片 图片源为 "风扇3.png"

当点击回调函数 button_click1 被触发
全局global 变量 flag
将变量 flag 赋值为 0

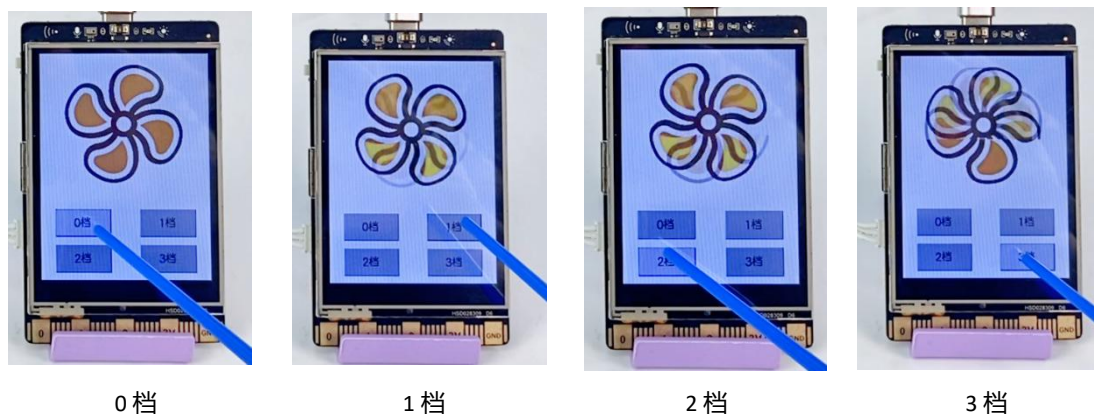
当点击回调函数 button_click2 被触发
全局global 变量 flag
将变量 flag 赋值为 1

当点击回调函数 button_click3 被触发
全局global 变量 flag
将变量 flag 赋值为 2

当点击回调函数 button_click4 被触发
全局global 变量 flag
将变量 flag 赋值为 3
    
```

4. 程序运行

点击右上角的运行按钮，程序运行成功后，依次点击界面上的 0~3 档的按钮，观察行空板屏幕上的扇叶是否有这些变化：0 档时，扇叶不会转动；1 档时，扇叶转动得比较慢；2 档时，扇叶转动得较快；3 档时，扇叶转动得最快。



5. 试一试

根据上面教的方法，尝试着用几张连续动作的图片做一部翻书动画。

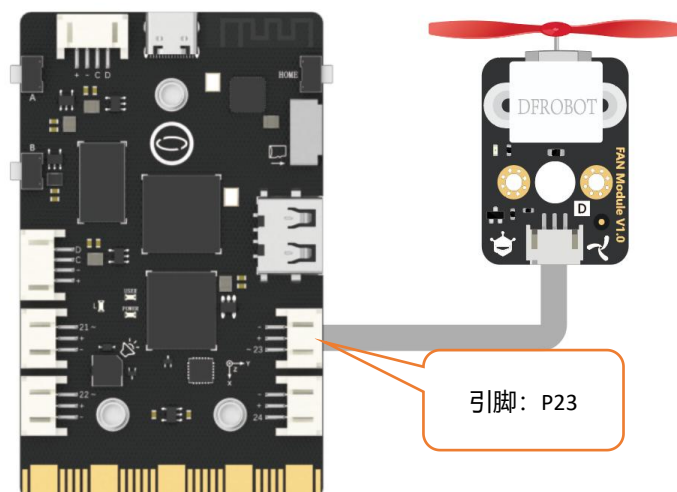
任务三：使用 PWM 信号控制风扇模块的转速

通过任务二，我们已经设计好不同档位下风扇转动的动画效果，接下来，我们将接入硬件——风扇模块，学习用 PWM 信号控制风扇的速度，实现变速风扇的功能。

1. 硬件连接

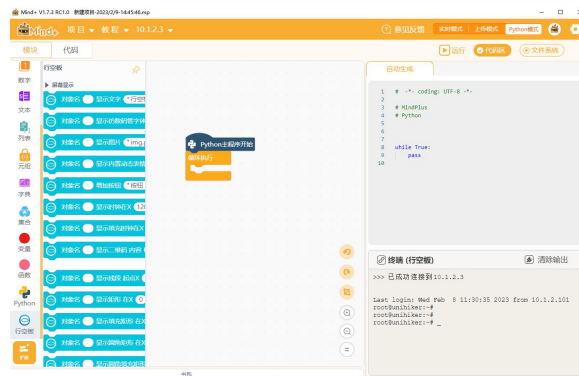
使用连接线将风扇模块连接到行空板上的 **P23 脚**，如下图。硬件连接成功后，使用 USB 线将行空板连接到计算机。

注意：[关于“风扇模块”的相关介绍，见知识园地。](#)



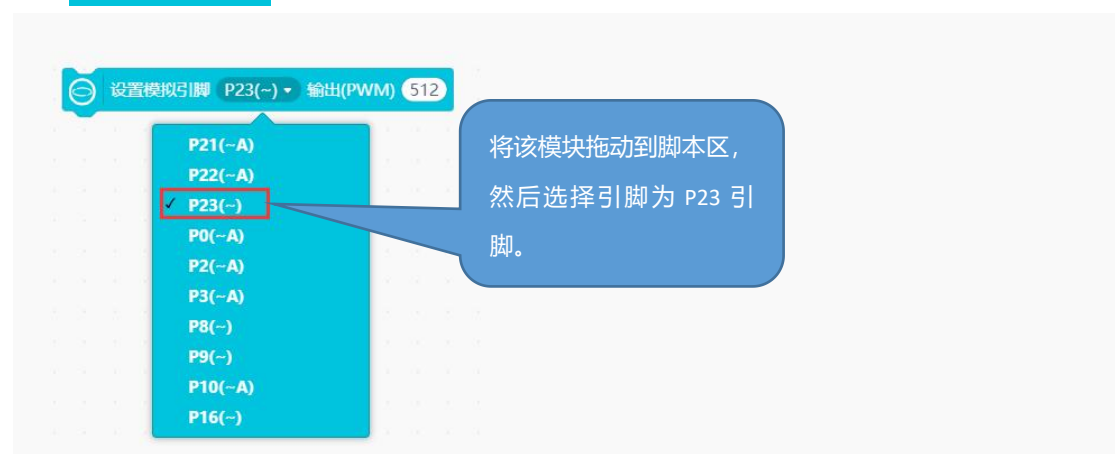
2. 软件准备

按照任务一的方法准备好软件。



3. 编写程序

- (1) 因为项目中使用的风扇模块可以直接用行空板里的引脚操作指令来控制，所以使用“设置模拟引脚 () 输出 (PWM) (512)”指令来驱动风扇模块即可。



- (2) 引脚设置好之后，再复制 3 个一模一样的指令，并将这 4 个指令的 PWM 值 (风速) 按照 0,400,700,1023 的这几个数设置。



- (3) 设置好每个档位的转速后，分别将它们拖动到任务二对应判断条件里的执行结果中去。例如：0 档的风速代码，则拖动到判断条件为 flag=0 的执行结果中。

Python主程序开始

对象名 风扇图片 显示图片 "风扇1.png" 在X 120 Y 100

更新对象名 风扇图片 的基准点为 中心

对象名 增加按钮 "0档" 在X 20 Y 220 宽 80 高 40 点击回调函数 button_click1

对象名 增加按钮 "1档" 在X 140 Y 220 宽 80 高 40 点击回调函数 button_click2

对象名 增加按钮 "2档" 在X 20 Y 270 宽 80 高 40 点击回调函数 button_click3

对象名 增加按钮 "3档" 在X 140 Y 270 宽 80 高 40 点击回调函数 button_click4

将变量 flag 赋值为 0

循环执行

如果 变量 flag = 0 那么执行

更新图片对象 风扇图片 图片源为 "风扇1.png"

设置模拟引脚 P23(-) 输出(PWM) 0

如果 变量 flag = 1 那么执行

等待 0.1 秒

更新图片对象 风扇图片 图片源为 "风扇1.png"

等待 0.1 秒

更新图片对象 风扇图片 图片源为 "风扇2.png"

等待 0.1 秒

更新图片对象 风扇图片 图片源为 "风扇3.png"

设置模拟引脚 P23(-) 输出(PWM) 400

如果 变量 flag = 2 那么执行

等待 0.05 秒

更新图片对象 风扇图片 图片源为 "风扇1.png"

等待 0.05 秒

更新图片对象 风扇图片 图片源为 "风扇2.png"

等待 0.05 秒

更新图片对象 风扇图片 图片源为 "风扇3.png"

设置模拟引脚 P23(-) 输出(PWM) 700

如果 变量 flag = 3 那么执行

等待 0.01 秒

更新图片对象 风扇图片 图片源为 "风扇1.png"

等待 0.01 秒

更新图片对象 风扇图片 图片源为 "风扇2.png"

等待 0.01 秒

更新图片对象 风扇图片 图片源为 "风扇3.png"

设置模拟引脚 P23(-) 输出(PWM) 1023

当点击回调函数 button_click1 被触发

全局global 变量 flag

将变量 flag 赋值为 0

当点击回调函数 button_click2 被触发

全局global 变量 flag

将变量 flag 赋值为 1

当点击回调函数 button_click3 被触发

全局global 变量 flag

将变量 flag 赋值为 2

当点击回调函数 button_click4 被触发

全局global 变量 flag

将变量 flag 赋值为 3

- (4) 同学们是否发现 1 档, 2 档, 3 档的动画程序只是等待时间不一样。那当一段程序需要重复使用时, 怎样做才能提高效率呢? 有部分同学可能会提到将这段程序封装成函数, 使用的时候直接调用这个函数就行了, 但是这三段程序执行的等待时间不一样, 如何才能让一个函数执行 3 种不同的等待时间呢?

其实想要实现这个功能不难，现在为大家介绍一个新的函数指令——带参函数指令。具体的使用方法如下图。

[注意：关于“带参函数”的相关介绍，见知识园地。](#)

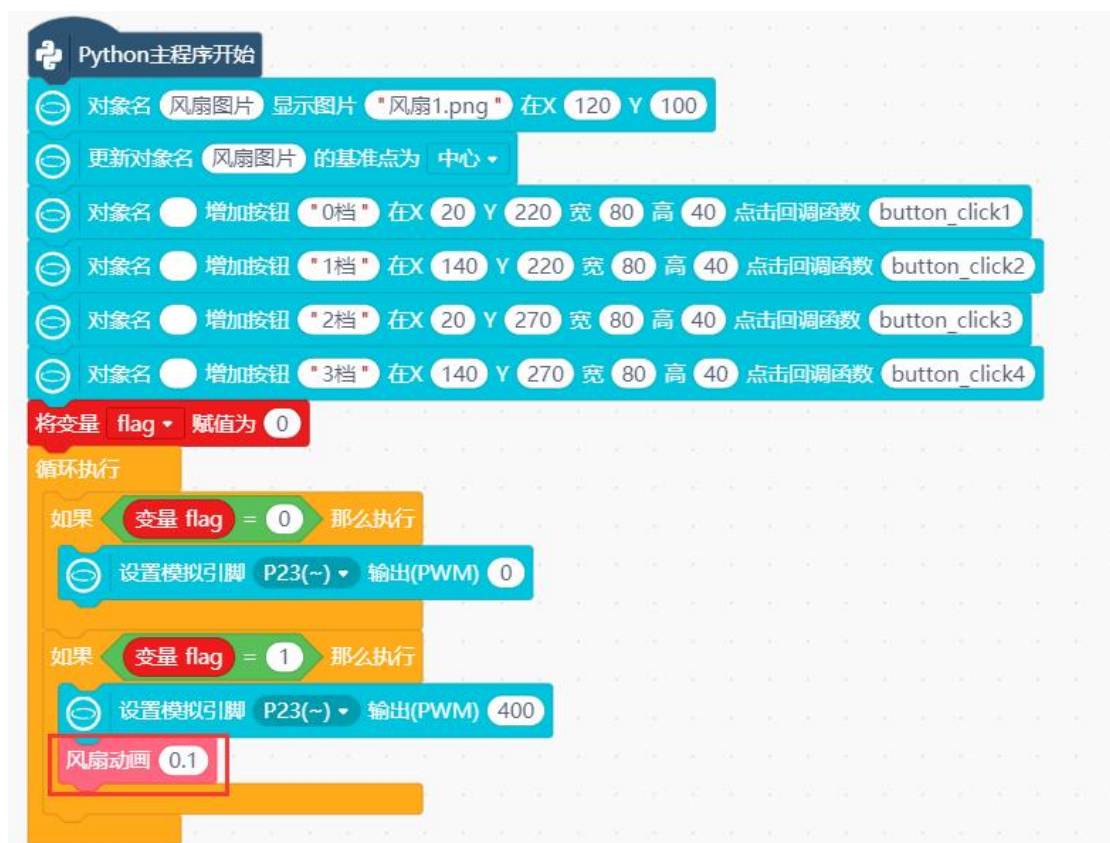
1. 点击模块中的“函数”，然后点击“自定义模块”。
2. 修改函数名为：风扇动画。
3. 点击“添加输入项”。
4. 并将参数名称修改为：时间，点击完成，完成带参函数的创建。



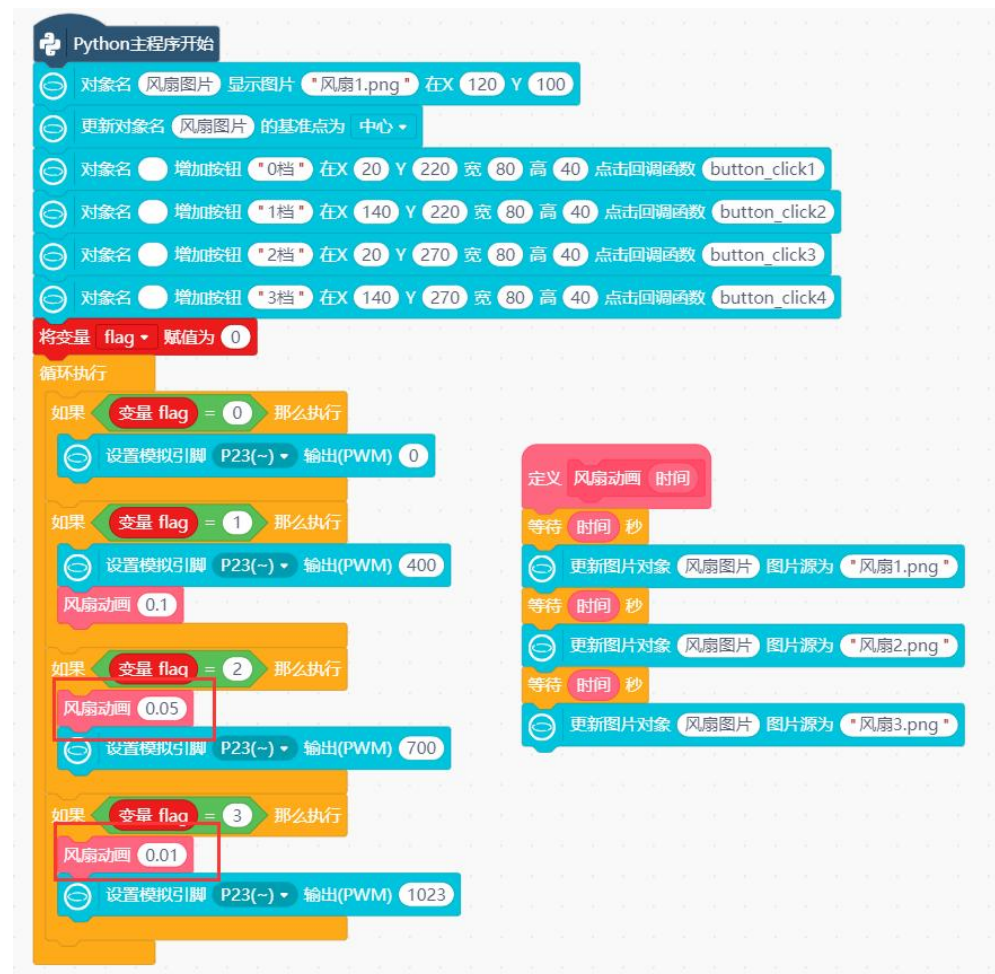
- (5) 将动画效果的程序放在函数下，并将函数参数名——时间拖动到等待（）秒的模块中。



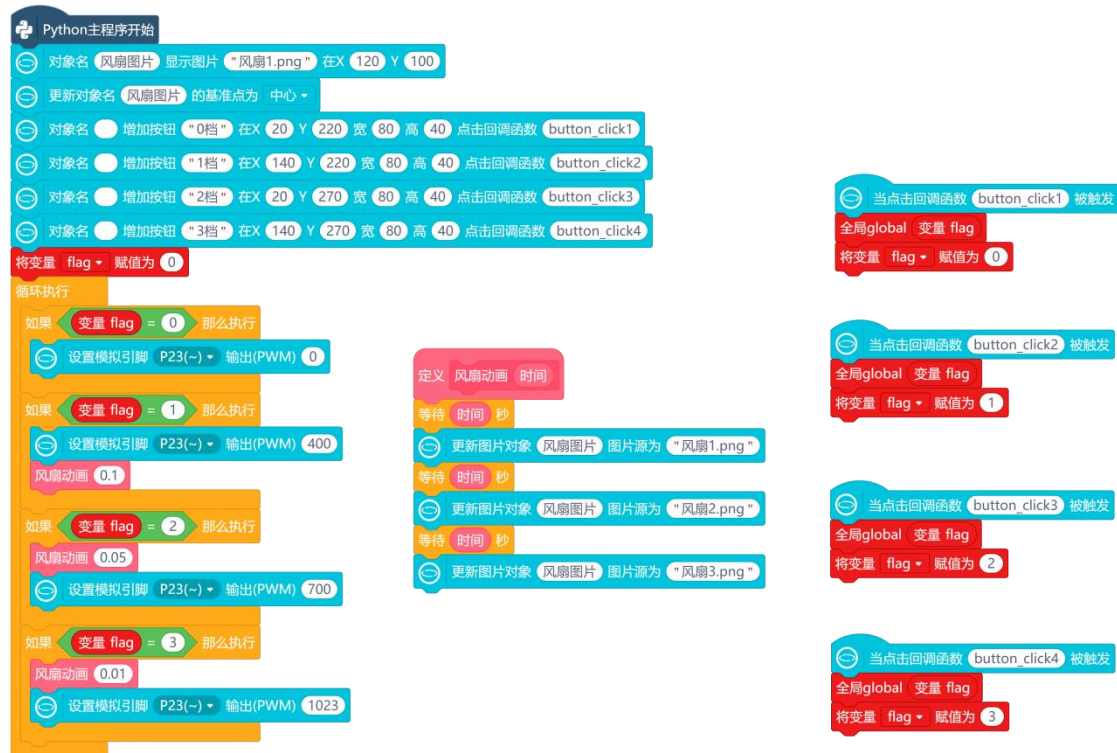
(6) 在调用风扇动画的带参函数时，将等待的时长，填写到调用函数的参数中即可。



(7) 如上一步，当按下 1 档的按钮后，每张图片切换的时间为 0.1 秒，所以将风扇动画函数的参数时间设置为 0.1，那么利用相同的方法，分别设置 2 档和 3 档的时间为 0.05 和 0.01。



4. 参考程序



5. 程序运行

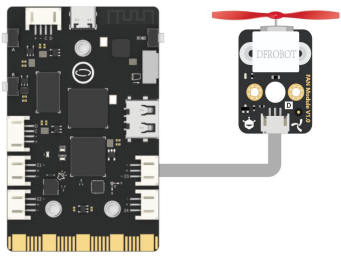
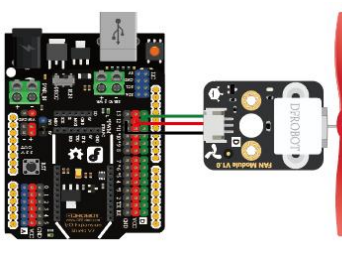
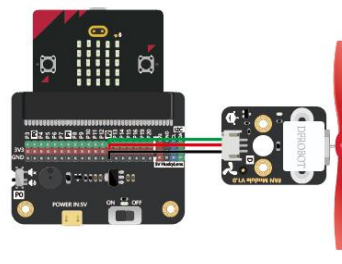
点击右上角的运行按钮，程序运行成功后，依次点击界面上的 0~3 档的按钮，观察风扇模块转动的情况。

关闭风扇	
轻风	
微风	
强风	

知识园地

1. 认识风扇模块

这节课我们使用的**输出元素**是风扇模块。为了解决因端口电流太弱无法驱动电机问题，该风扇模块将电机驱动与直流电机一起集成在模块上了，实现了**直接插在端口上**就能使用。

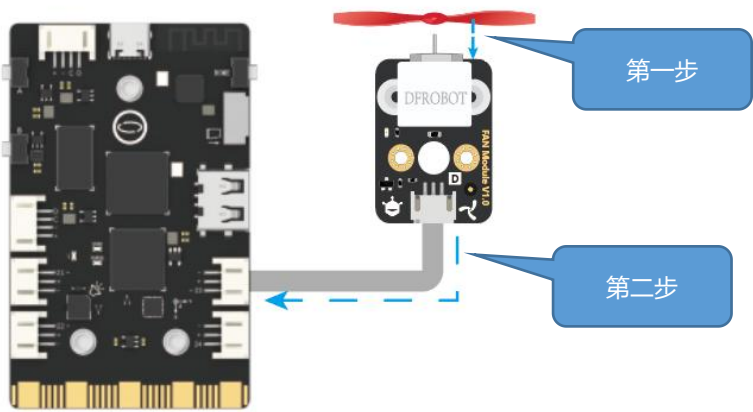
		
行空板 连接方式	Arduino 连接方式	Micro:bit 连接方式

1) 风扇模块工作原理

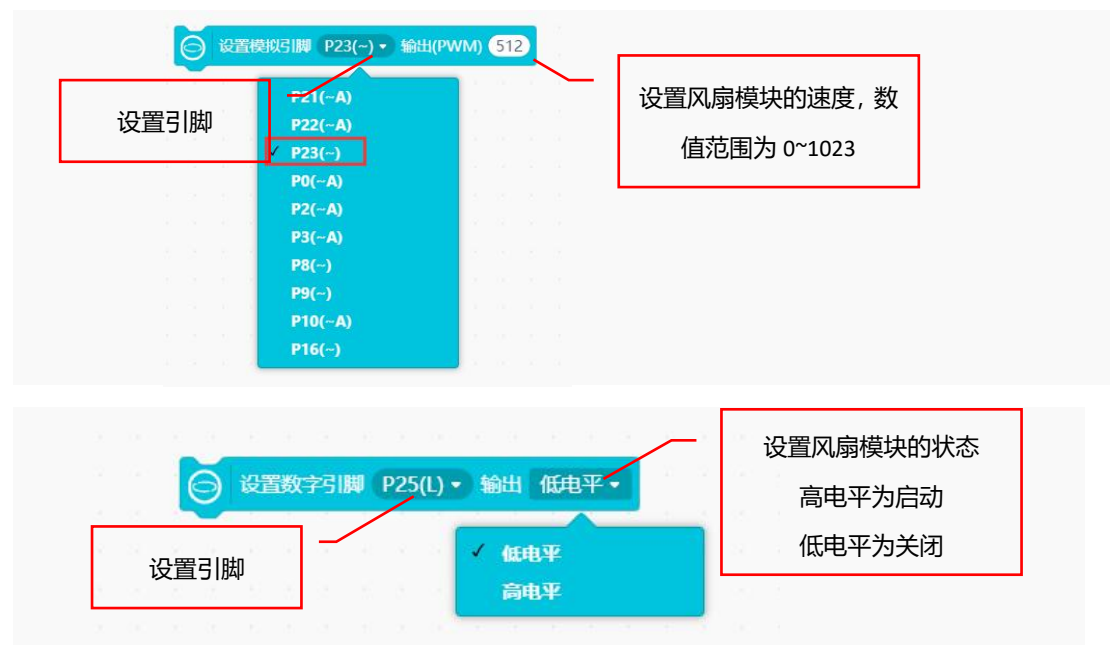
在过程控制系统中，风扇模块主要用来接收控制器送来的控制信号，当它接收到的信号为数字信号（0 或 1）时，执行的是开或关这两个动作；当它接收到的信号为 PWM 信号时，执行的是随着 PWM 信号的增强，电机的转速不一样。

2) 风扇模块使用方法

前面我们讲了风扇模块可以直接插在扩展板或者控制器的端口上使用，使用方法见下图，首先将扇叶插在模块上，然后将整个模块连接到控制器或者扩展板上。

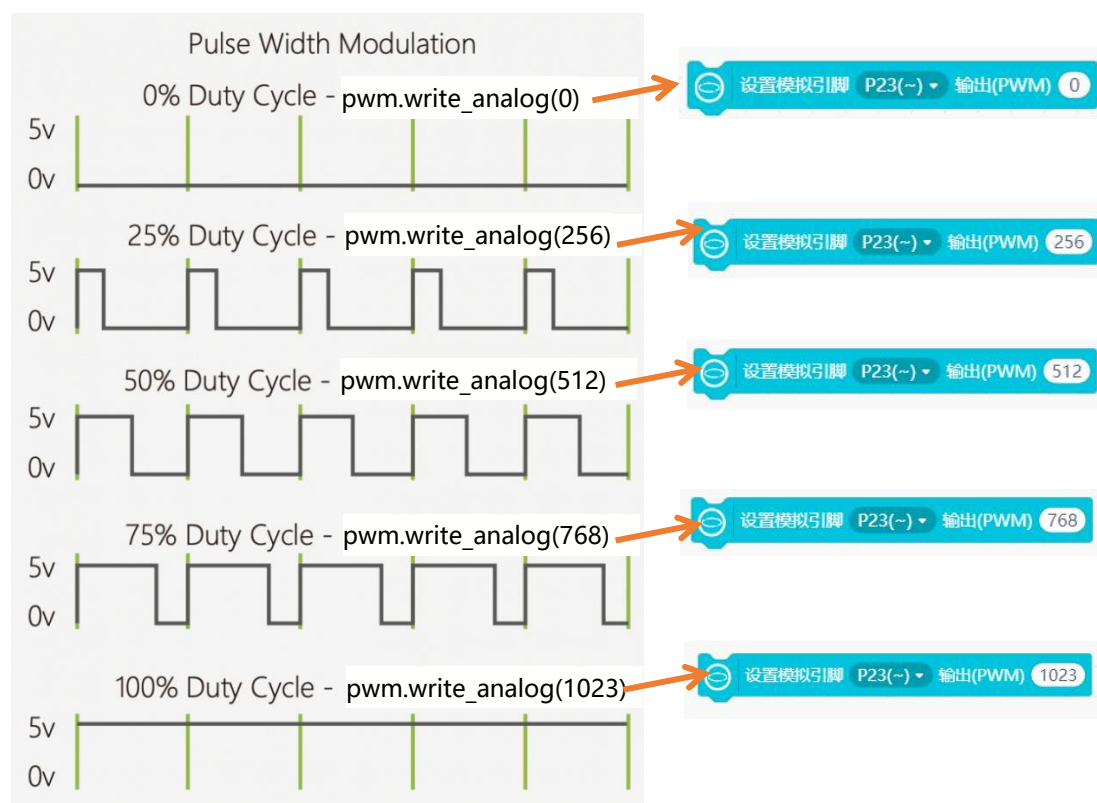


在 mind+中，我们可以利用“**设置模拟引脚 () 输出 (PWM) (512)**”指令和“**设置数字引脚 () 输出 (低电平)**”指令来控制风扇模块。



2. 认识 PWM 信号输出

PWM 是一项通过数字方法来获得模拟量的技术。它一般运用于**电机速度控制、LED 灯亮度调节、音频放大器**等。PWM 信号是用数字控制来形成一个方波，方波信号只有开关两种状态（也就是我们数字引脚的高低），然后通过控制开与关所持续时间的比值就能模拟到一个 0 到 5V 之间变化的电压。开（学术上称为高电平）所占用的时间就叫做脉冲宽度，所以 PWM 也叫做脉冲宽度调制。接下来，通过下面五个方波来更形象的了解一下 PWM。



上图绿色竖线代表方波的一个周期。每个输出(PWM)(xxx) 中写入的值都能对应一个百分比，这个百分比也称为占空比(Duty Cycle)，指的是一个周期内高电平持续时间比上低电平持续时间得到的百分比。图中，从上往下，第一个方波，占空比为 0%，对应的值为 0。电机转动的速度最小，也就是停止的状态。高电平持续时间越长，也就越快。所以，最后一个占空比为 100%的对应值是 1023，电机转动的速度最快。50%就是最快速度的一半了，25%则相对较慢。

3. 认识带参函数

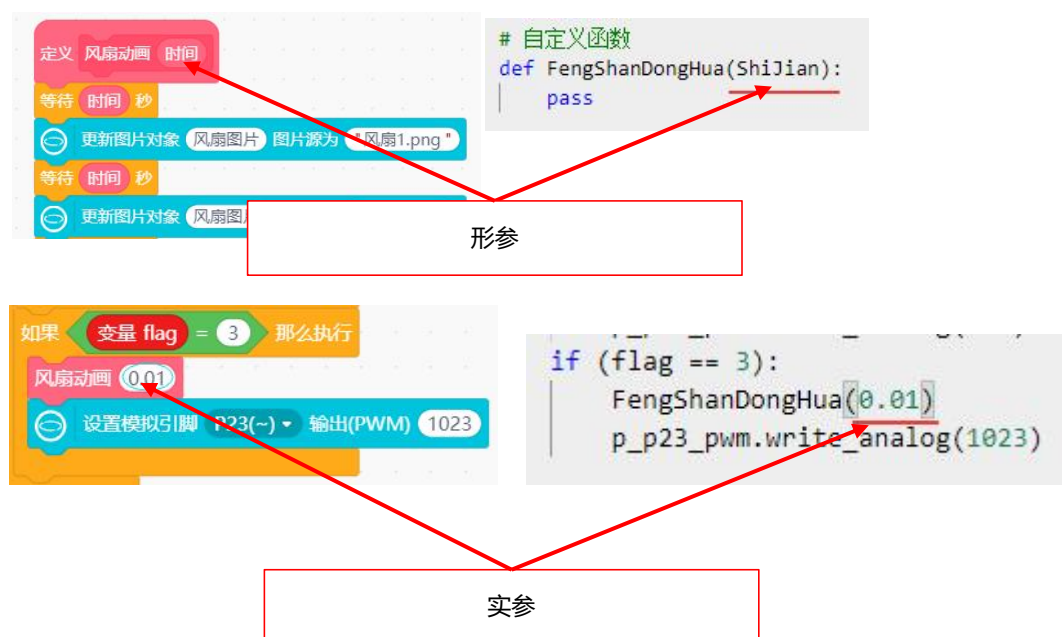
在了解带参函数前，我们先回顾一下什么是函数，函数，就是将一段经常使用的程序封装起来，在需要使用这个程序的位置调用这个函数即可，并且这个函数可以多次调用，这样做可以减少重复代码。而带参函数就是带有参数的函数。参数，也叫参变量，是一个变量。带参函数，就是在创建函数时，在函数后面同时新建了一个变量。例如任务三程序中，时间为风扇动画的参变量。



在调用带参函数时，因为函数定义了参数，因此调用这个函数时，需要写入具体的参数值，这个过程就称为传入参数。



在函数里，一般有两个参数，形参和实参，而他们的区别就在于，函数名后面的参数时长称为形参，而调用函数的时候传入的具体参数值叫做实参。形参和实参的关系是，形参是用来接收实参的。因此，在调用带参函数时，将实参修改为 0.1、0.05、0.01 就可以控制电机的转速了。



4. 什么是全局变量

变量分为局部变量和全局变量，局部变量就是我们平时常用的变量，只能在它所在的函数或循环中使用，并且随函数或循环结束而结束，而全局变量能在其他函数里或循环中使用该变量的值，一直到程序结束。将变量放入 **全局 global** 指令中，就能定义该变量为全局变量。



全局变量一般在什么地方用呢？在任务二的程序中，每个“当点击回调函数（button_click1）被触发”的指令下都将变量 flag 定义为全局变量了，因为在这几个地方修改了变量的值，需要给变量 flag 赋一个新的值，因此每当这个回调函数被触发时，都需要将变量 flag 定义为全局变量。



5. 指令学习

动手实践里主要使用了以下操作指令。

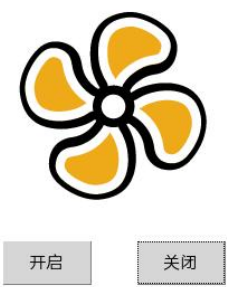
Block	功能说明
	该指令用来设置 PWM 信号的输出值，范围为 0~1023，数值越大转速越快。
	该指令用于在行空板屏幕显示按钮对象，在指令中可以设置对象名、按钮上显示文字、按钮位置、尺寸和按钮功能实现的回调函数名。
	该指令用于定义一个带参函数，函数名为：风扇动画，形参名为：时间。

	该指令利用函数名调用函数，并在调用的过程中传入实际参数值。
	比较运算符“=”，需要两个操作数，当两个操作数的值相等时，返回值为 true，否则为 false。
	全局变量，将变量放入这个模块中，就定义该变量为全局变量。

挑战自我

在知识园地里介绍了风扇模块的另一种使用方法。接下来，请使用“设置数字引脚（）输出（低电平）”的指令修改任务三的程序，让变速风扇变成定速风扇。

提示：风扇模块被打开时，行空板屏幕上的动画也要动起来。



第 5 课 美食料理机

以前的人们吃东西只注重饱腹，现在的人们吃东西会追求味道、健康。而随着科技的发展，市面上也出现了一些新的厨电产品，它们能帮助人们得到更加健康的生活方式，比如破壁机、榨汁机等料理机。因为它们的超高转速可以搅碎食物，使食物的营养元素释放出来，是现代居家必备的产品。现在市面上的料理机能制作的食物也越来越多，比如豆浆、果汁、米糊、浓汤等等。接下来，我们就利用行空板、角度传感器与风扇搭建一款可调节不同菜单模式的美食料理机。



任务目标

在行空板屏幕上设置一个开始按钮，当按下开始按钮后，进入菜单（豆浆，果蔬，浓汤）选择界面，然后旋转角度传感器切换菜单。例如：当需要制作的食物是豆浆，我们旋转角度传感器将菜单切换到豆浆，RGB 灯显示白色，然后点击豆浆，启动风扇，等几秒过后，风扇停止，屏幕上显示制作完成的信息。

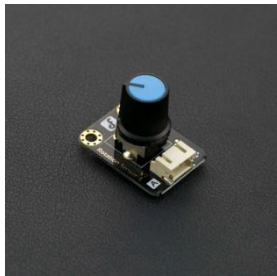


知识点

1. 了解模拟信号的概念。
2. 掌握行空板获取模拟输入信号的方法。
3. 掌握角度传感器的工作原理及使用方法。
4. 掌握在行空板上绘制圆形图案和更新文字。

材料清单

硬件清单:

		
行空板 x1	风扇模块 x1	角度传感器 x1
		
RGB 灯带 x1	白色硅胶绞线 x1	Type-C USB 线 x1

软件使用: Mind+编程软件 x1

下载地址: <https://mindplus.cc/>

动手实践

美食料理机，主要是模拟破壁机的工作原理，通过旋转角度传感器选择菜单，而且每种菜单下，风扇模块工作的时间不一样。接下来，通过三个简单的小任务来学习如何制作一个美食料理机。

任务一：设计美食料理机的控制界面

在此项任务中，利用填充圆形指令和回调函数设计美食料理界的控制界面。

任务二：设计菜单切换功能

在完成控制界面的设计后，我们加入角度传感器，通过它来选择菜单（豆浆/果蔬/浓汤）。

任务三：制作美食料理机

在任务二的基础上，加上风扇与 RGB 灯带模块，当点击不同的菜单时，风扇模块工作的时长不一样，RGB 灯模块呈现的效果也不一样。例如豆浆模式下，RGB 灯显示白色，风扇工作 15 秒；果蔬模式下，RGB 灯显示绿色，风扇工作 10 秒；浓汤模式下，RGB 灯显示红色，风扇工作 5 秒。

任务一：设计美食料理机的控制界面

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。

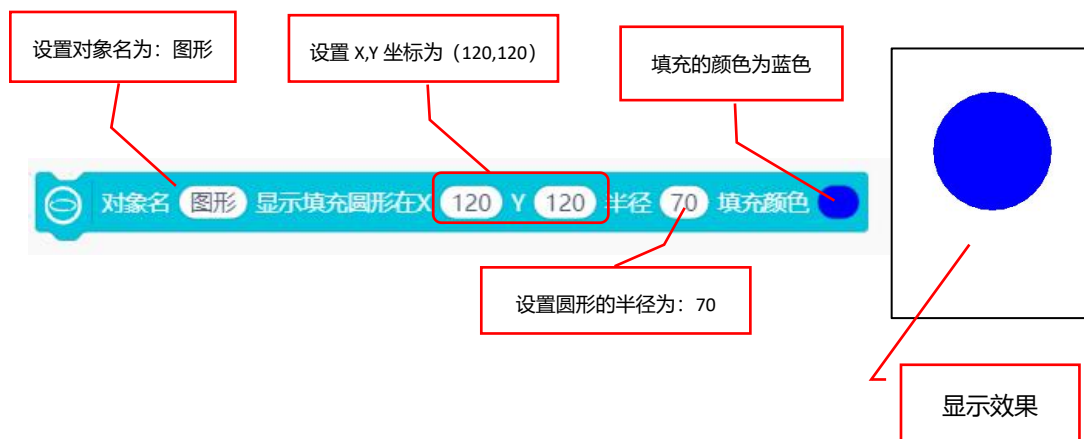


3. 编写程序

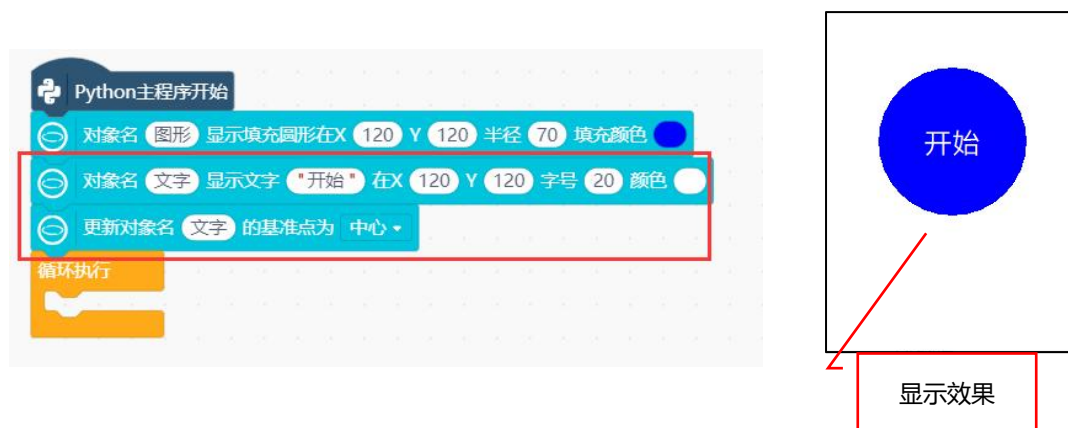
首先观察一下，控制界面上都包含哪些元素？一个圆形的开始按钮，效果如下图。



- (1) 设计控制界面，首先要将“对象名 () 显示填充圆形在 X(120)Y(100)半径(50)填充颜色 蓝”指令拖动到脚本区。设置对象名为“图形”，然后根据实际情况设置合适的坐标与半径。



- (2) 圆形的图案绘制好之后，利用“对象名 () 显示文字 (" ") 在 X(0)Y(0) 字号 () 颜色 蓝”和“更新对象名 () 的基准点为 (中心)”指令，让文字“开始”显示在蓝色圆形图案的中间。



- (3) 因为整个项目的功能是需要点击“开始”按钮后，才能进入菜单模式，所以我们要使用“对象名 () 的点击回调函数为 ()”和“当点击回调函数 () 被触发”指令，将“开始”这个文字设置成一个按钮。



- (4) 在“当点击回调函数 (开始) 被触发”指令下添加一句“打印 (" ")”指令，并在里面填写“开始”的文字。主要用于验证触发按钮是否设置成功。

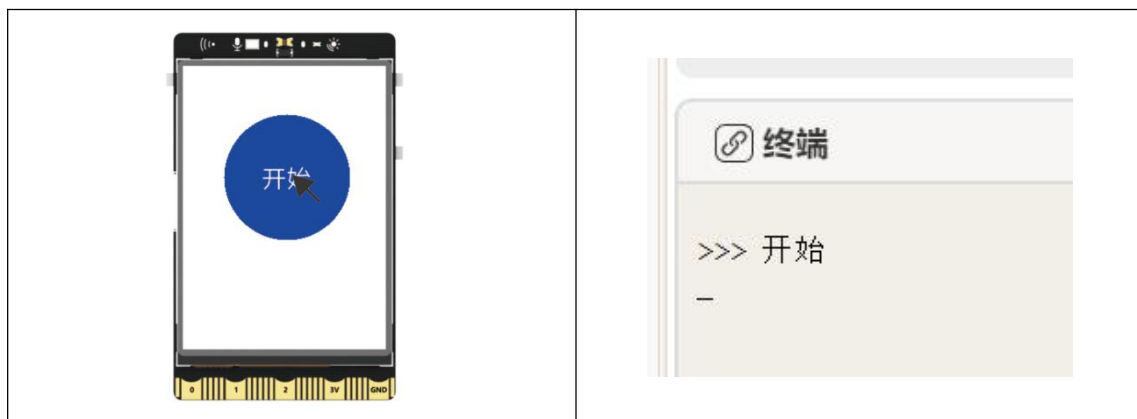


4. 参考程序



5. 程序运行

检查行空板是否已连接到 mind+, 无误后, 点击右上角的运行按钮, 程序运行成功后, 行空板屏幕上会显示如下表的界面, 然后点击“开始”位置, 观察终端区是否会显示“开始”的文字。



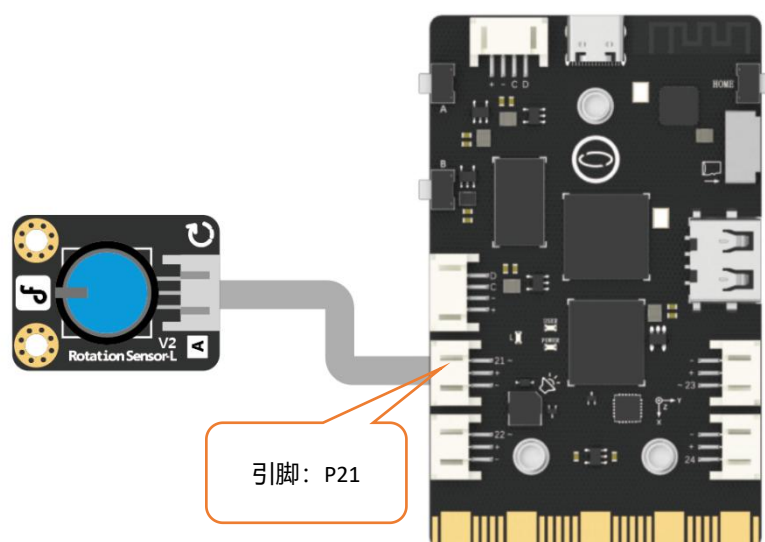
任务二：设计菜单切换功能

前面我们讲过, 当点击“开始”按钮后, 就进入菜单选择界面。接下来我们将接入硬件——角度传感器, 然后通过旋转角度传感器来切换菜单。

1. 硬件连接

使用连接线将角度传感器连接到行空板上的 P21 脚, 如下图。硬件连接成功后, 使用 USB 线将行空板连接到计算机, 然后按照任务一的方法准备好软件。

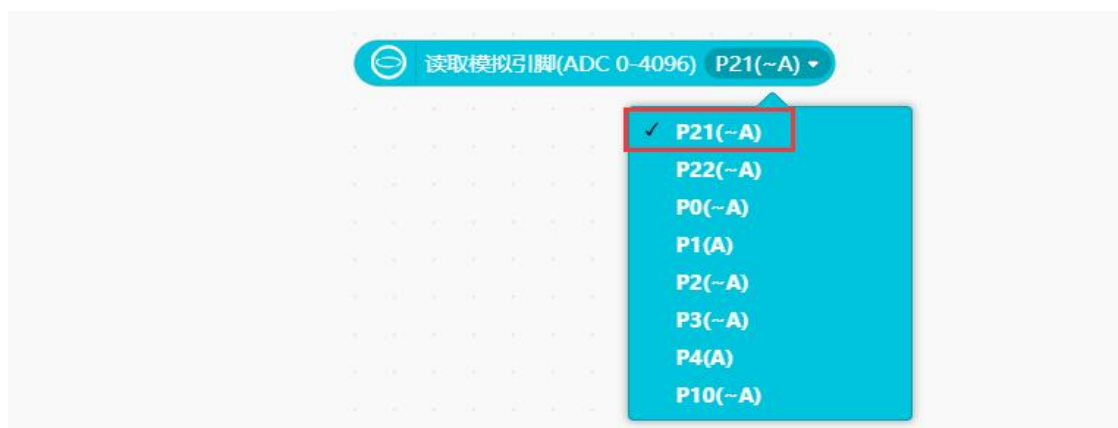
注意：关于“角度传感器”的相关介绍，见知识园地。



2. 编写程序

在程序开始之前，先分析一下需要几种菜单？通过观察家用破壁机的功能，一般豆浆，果蔬，浓汤菜单用得比较多，所以我们需要将角度传感器输出的值分成 3 个部分。接下来我们先学习如何读取角度传感器的值，然后再根据角度传感器的值区分每种菜单的范围值，最后利用条件判断语句来判断这三个范围值，当角度传感器输出的值达到指定范围，就执行对应模式下的结果。

- (1) 将“读取模拟引脚 (ADC 0~4096)”指令拖到脚本区，因为角度传感器是连接在 P21 引脚上的，所以指令里的引脚需要更改成 P21 引脚。



- (2) 使用“显示文字”与“更新文本内容”参数指令，将角度传感器检测到的模拟值显示在行空板上。



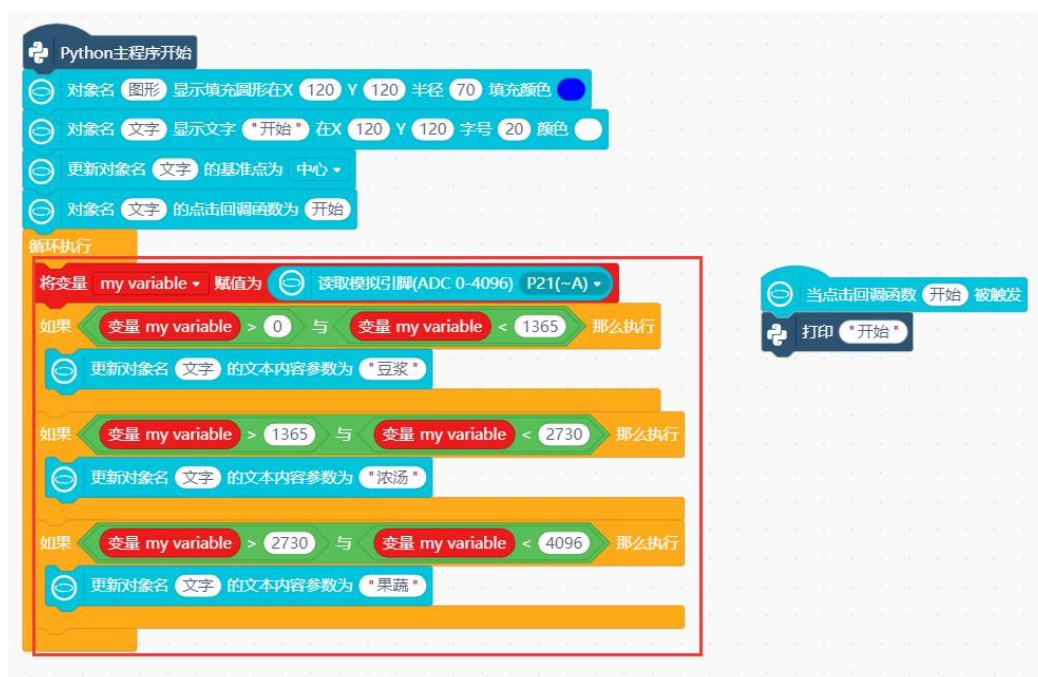
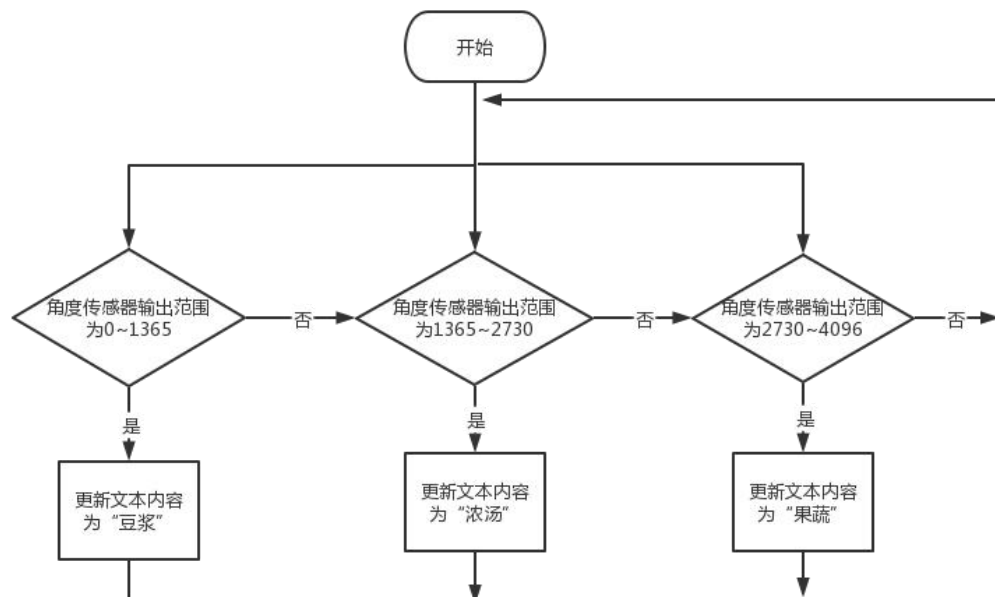
- (3) 根据“读取模拟引脚 (ADC 0~4096)”指令上的提示以及我们实际测试的范围值，角度传感器输出的模拟值范围为 0~4096。接下来我们将这个大范围值分成 3 个小范围值，用来区分不同的菜单模式。如下表。

菜单模式	豆浆	浓汤	果蔬
范围值	0~1365	1365~2730	2730~4096

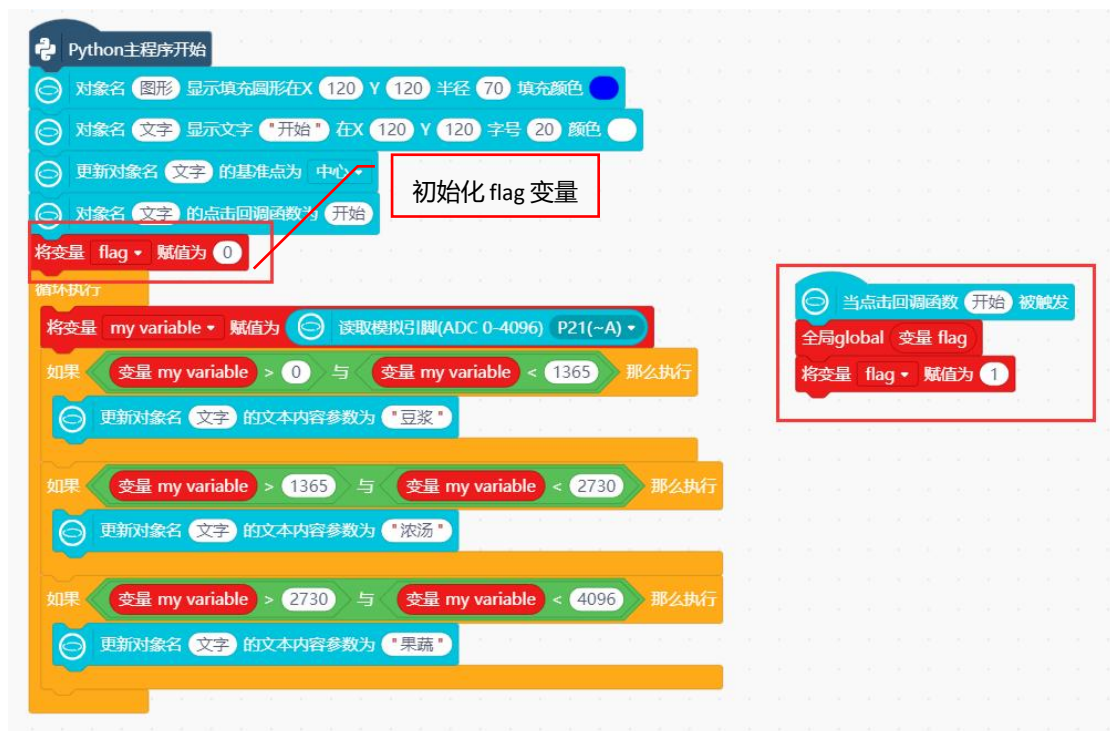
- (4) 在任务一的代码基础上，我们新建一个变量“my variable”，用来存储角度传感器输出的模拟值。



- (5) 在步骤 3 中，我们区分了不同菜单模式下的范围值，如豆浆的范围值为 0~1365，则当角度传感器输出的值在这个范围，就将屏幕上的文字更新成“豆浆”；那么同理，当范围值为 1365~2730 时，屏幕上的文字就更新成“浓汤”；范围值为 2730~4096 时，屏幕上的文字就更新成“果蔬”；执行流程如下。

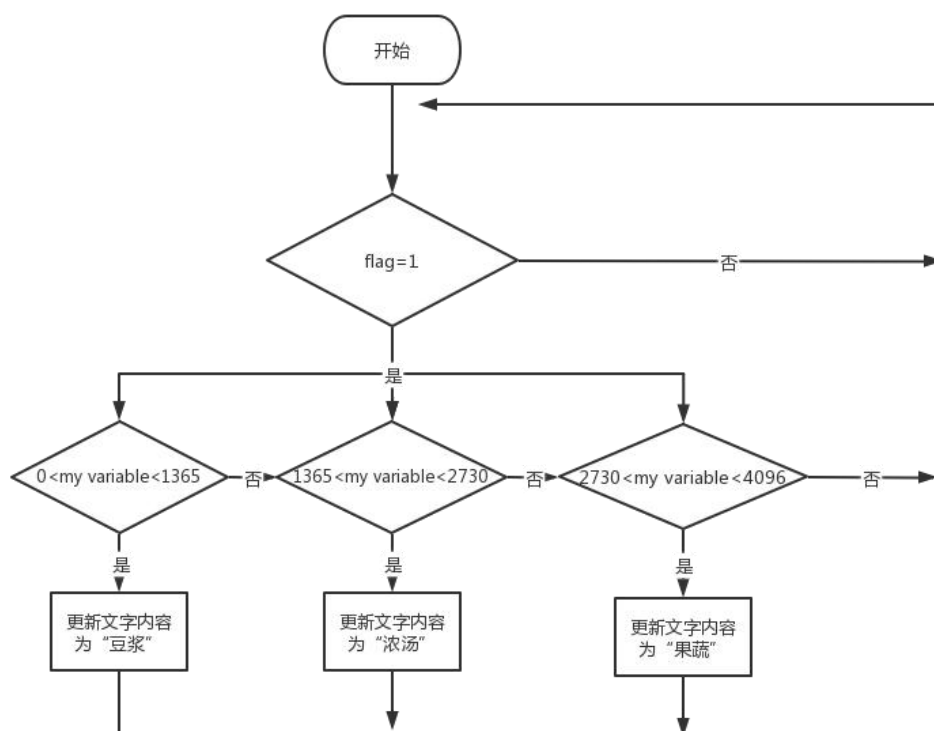


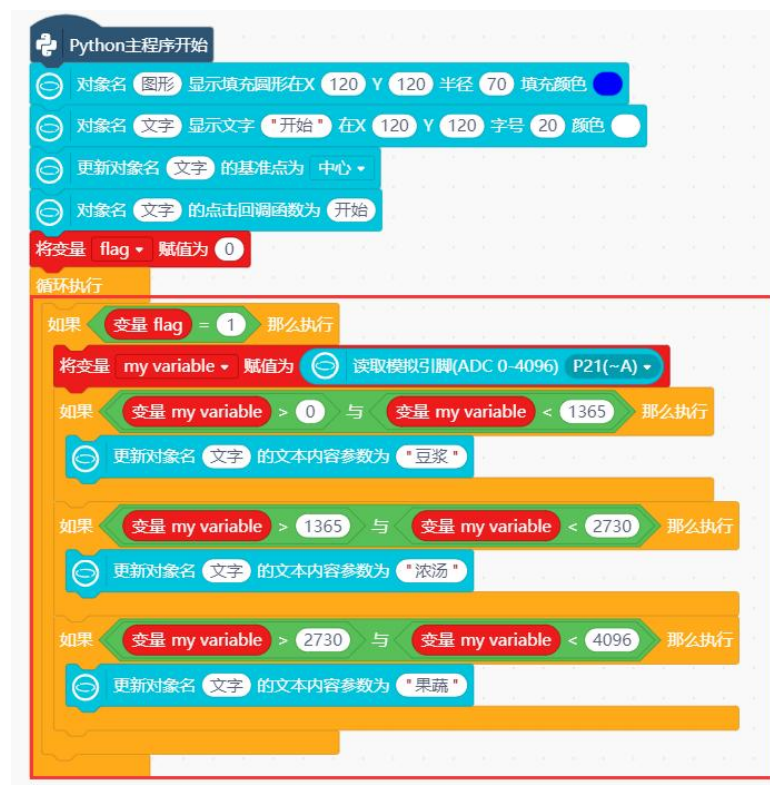
- (6) 因为项目设定的功能是，当点击开始按钮后，进入选择菜单的界面；所以我们需要新建一个标志变量 flag，当回调函数“开始”被点击后，就设置变量 flag 的值为 1。



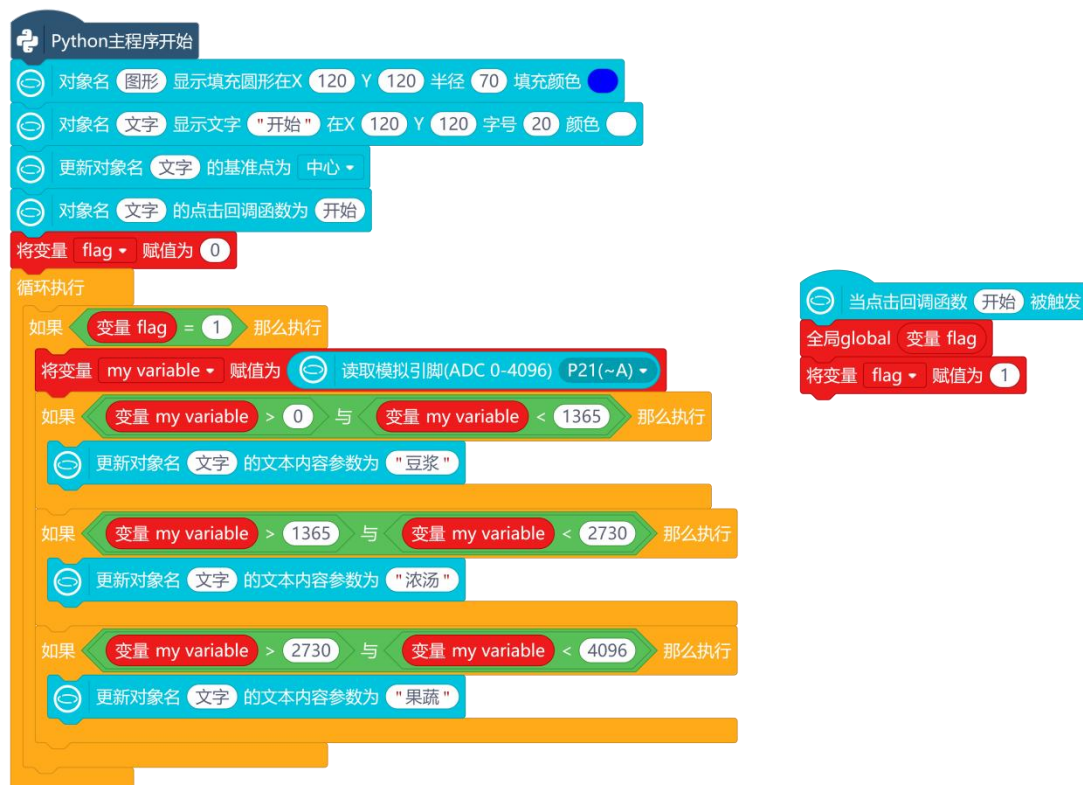
注意：需提前删除“开始”回调函数下的“打印(" ")”指令。

- (7) flag 变量设置好之后，我们需要在循环执行里面去判断这个变量的值，当 flag=1 时，就执行选择菜单程序。



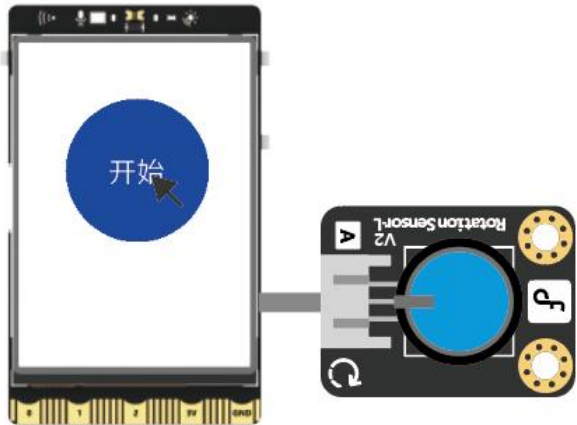
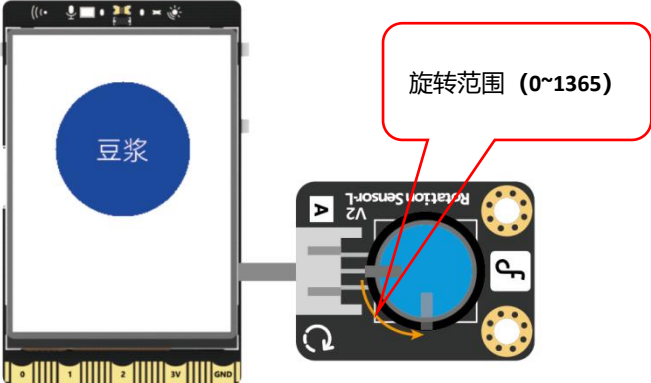
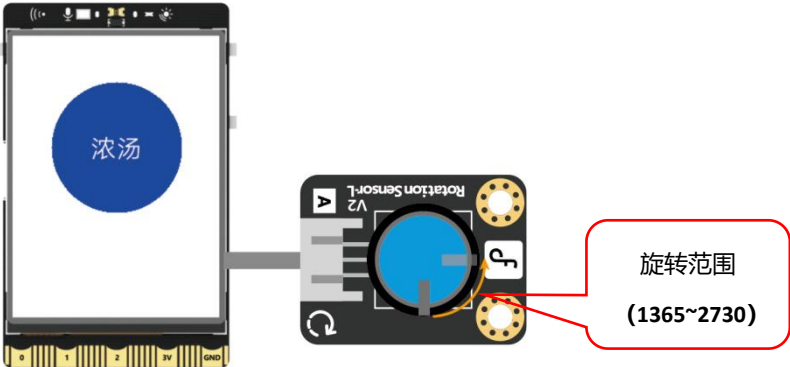
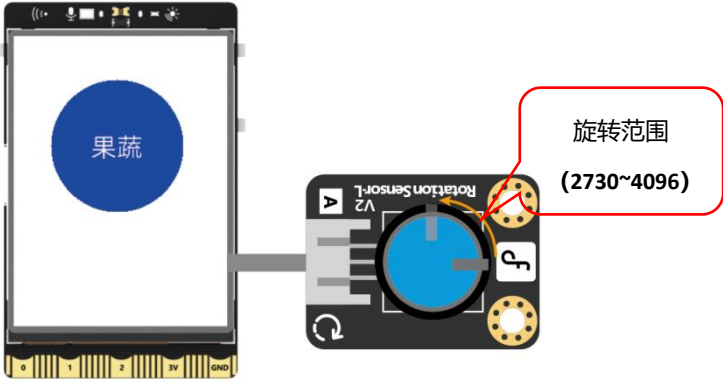


3. 参考程序



4. 程序运行

点击右上角的运行按钮，程序运行成功后，点击屏幕上的开始按钮进入菜单模式，然后旋转角度传感器选择菜单，最后点击取消按钮，观察界面是否回到初始界面。

初始界面	
豆浆模式	
浓汤模式	
果蔬模式	

5. 试一试

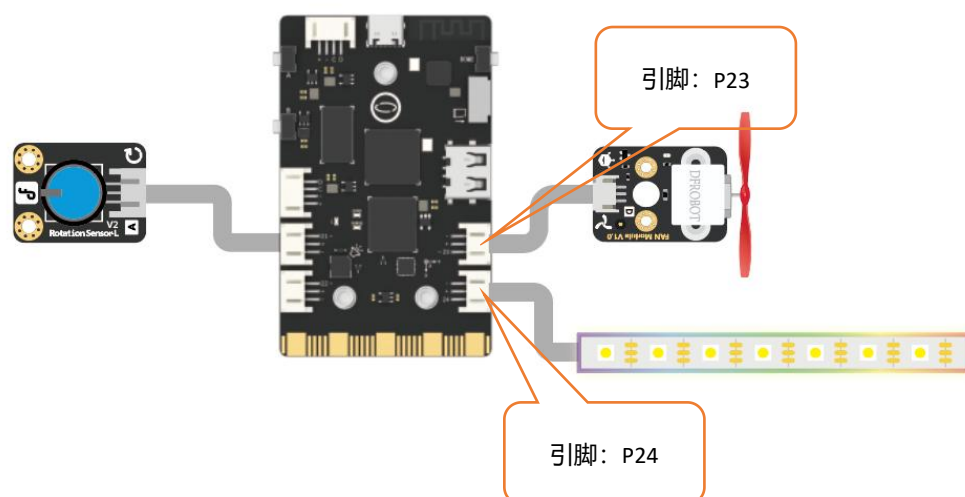
根据上面教的方法，尝试着再增加几种菜单模式，比如冰沙，米糊等。

任务三：制作美食料理机

通过任务二，我们已经完成了如何利用角度传感器切换菜单模式，接下来，我们将接入执行器——风扇模块和 RGB 灯，从而完成美食料理机的项目搭建。

1. 硬件连接

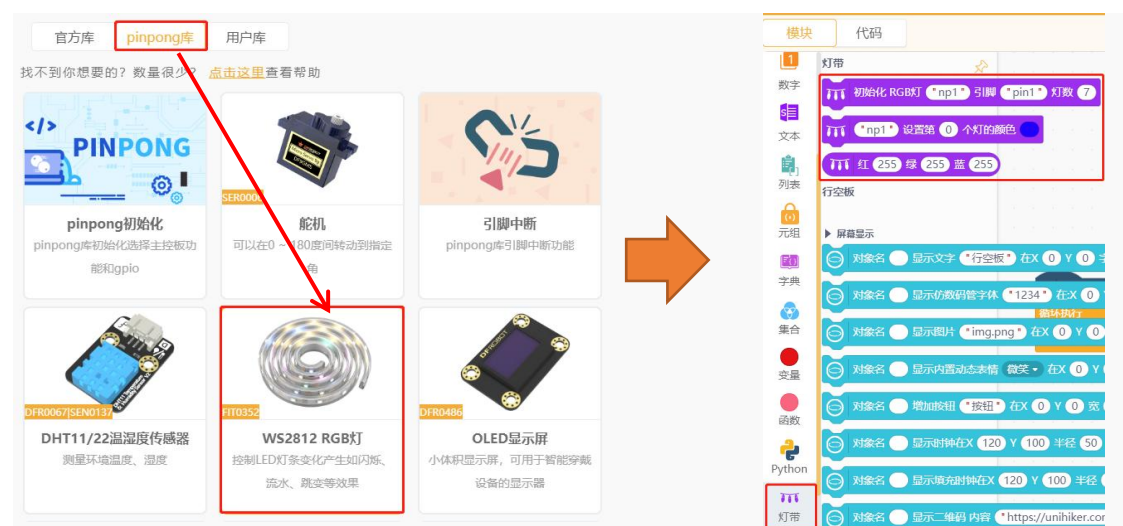
使用连接线将风扇模块和 RGB 灯带分别连接到行空板上的 P23 和 P24 脚，如下图。硬件连接成功后，使用 USB 线将行空板连接到计算机。



2. 软件准备

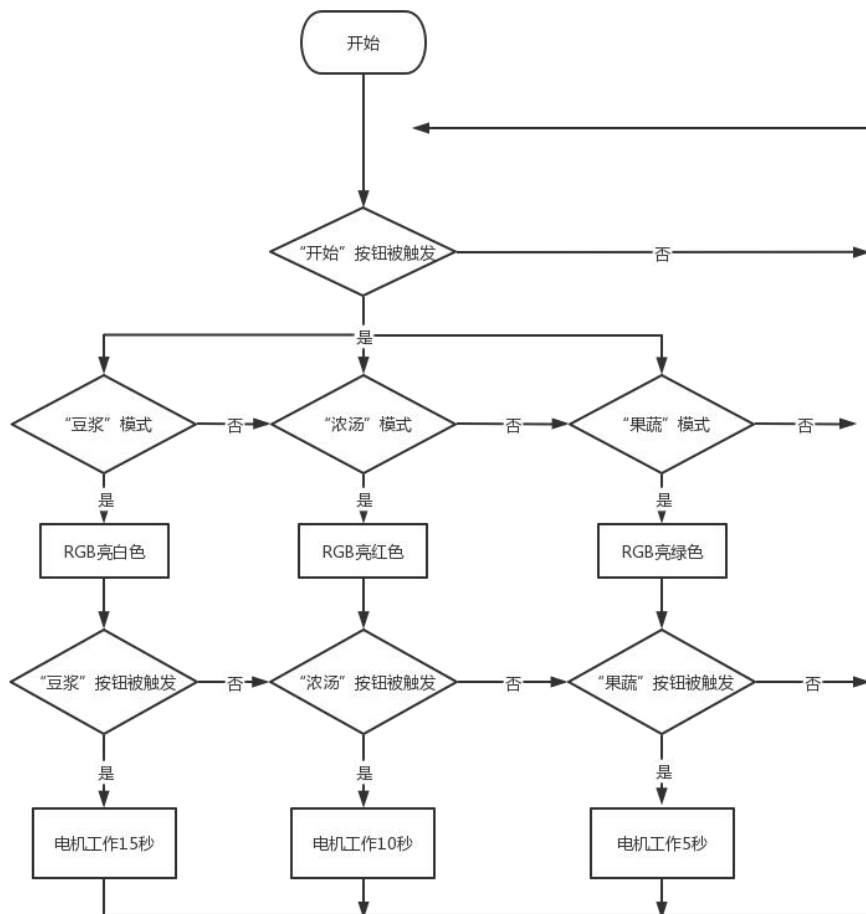
(1) 打开 Mind+, 按照任务一完成软件准备过程。

(2) 添加 WS2812 RGB 灯库，点击“扩展”，在 pinpong 库中找到“WS2812 RGB 灯”并点击，完成添加。

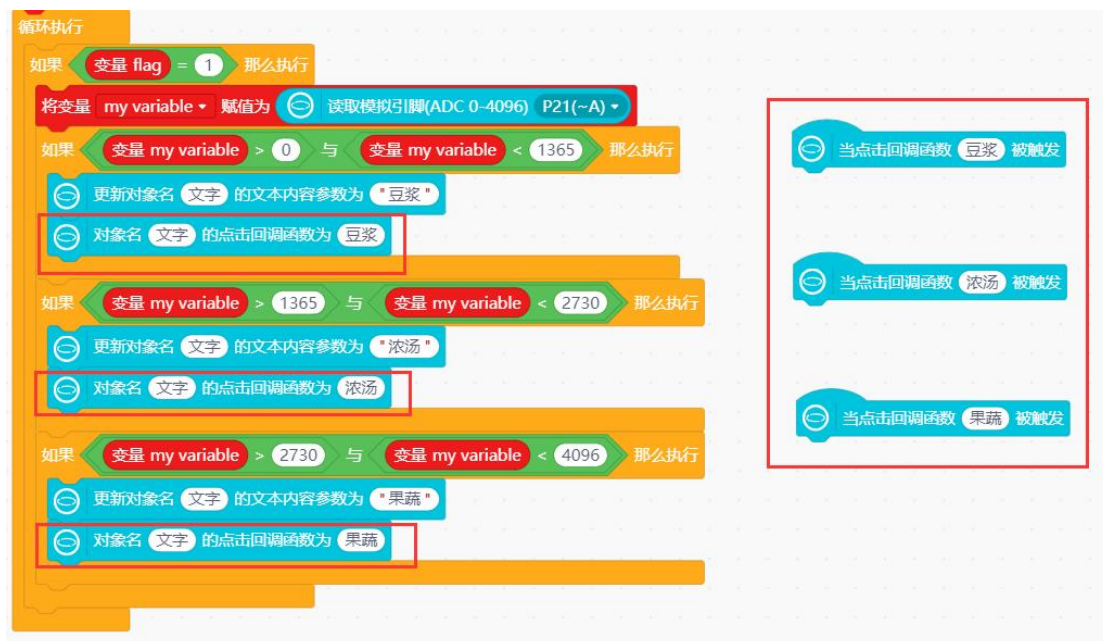


3. 编写程序

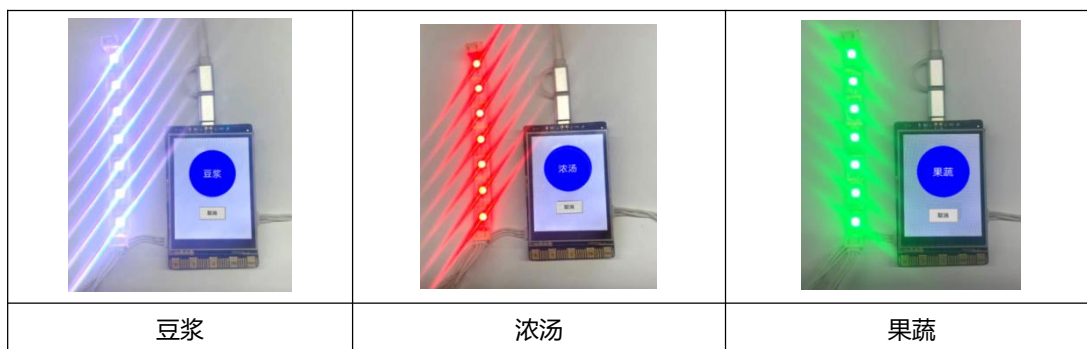
在编写程序前，先梳理一下任务三需要完成什么功能？在任务二中，我们已经完成了菜单切换和取消的功能，接下来我们需要为对应的菜单设置工作时长，比如当菜单切换至“豆浆”时，RGB 灯亮白色，点击“豆浆”按钮后，启动风扇工作 15 秒。整个项目的工作流程如下：



- (1) 由流程图得知，当用角度传感器切换菜单后，除了界面的文字会更新之外，还需要将更新后的文字生成一个点击回调函数。例如，当菜单为豆浆时，需要将“豆浆”这两个文字生成点击回调函数；菜单为浓汤或果蔬时，将“浓汤”或“果蔬”生成点击回调函数。



- (2) 从上面的流程图得知，当切换到对应的菜单后，RGB 灯也会显示不同的颜色，例如豆浆显示白色灯光；浓汤显示红色灯光；果蔬显示绿色灯光。实际效果如下图。



RGB 灯带与第三课的用法一样，首先在“Python 主程序开始”下面初始化 RGB 灯带。



因为这三种菜单模式下，只是显示得颜色不一样，所以我们可以使用带参函数，首先，我们新建一个函数名为“灯带”，参数名称为“颜色”的带参函数，然后设置一个变量“number”，使用 for 循环让每颗灯珠都显示同一种颜色，最后将参数名“颜色”拖动到“设置灯的颜色”模块中。



带参函数灯带设置好之后，在调用它时，只需要将显示的颜色填写到参数中即可，在项目中，我们通过使用“红 () 绿 () 蓝 ()”的指令来设置灯带的颜色。



注意：常用的 RGB 值，见知识园地。

- (3) 设置好灯带的显示效果后，接下来，就是控制电机了，根据流程图的分析，每种菜单模式下，电机工作的时间不一样。例如：当“豆浆”按钮被按下，电机转动 15 秒后停止，并在屏幕上显示“制作完成”；“浓汤”模式下，电机转动 10 秒后停止，并在屏幕上显示“制作完成”；“果蔬”模式下，电机转动 5 秒后停止，并在屏幕上显示“制作完成”。

注意：当菜单模式下的按钮被点击后，需要将标志变量 flag 的值更改为 0，让该菜单模式锁死，不然旋转角度传感器时，还是会切换到其他模式，而屏幕上的文字内容也不会更新成“制作完成”。

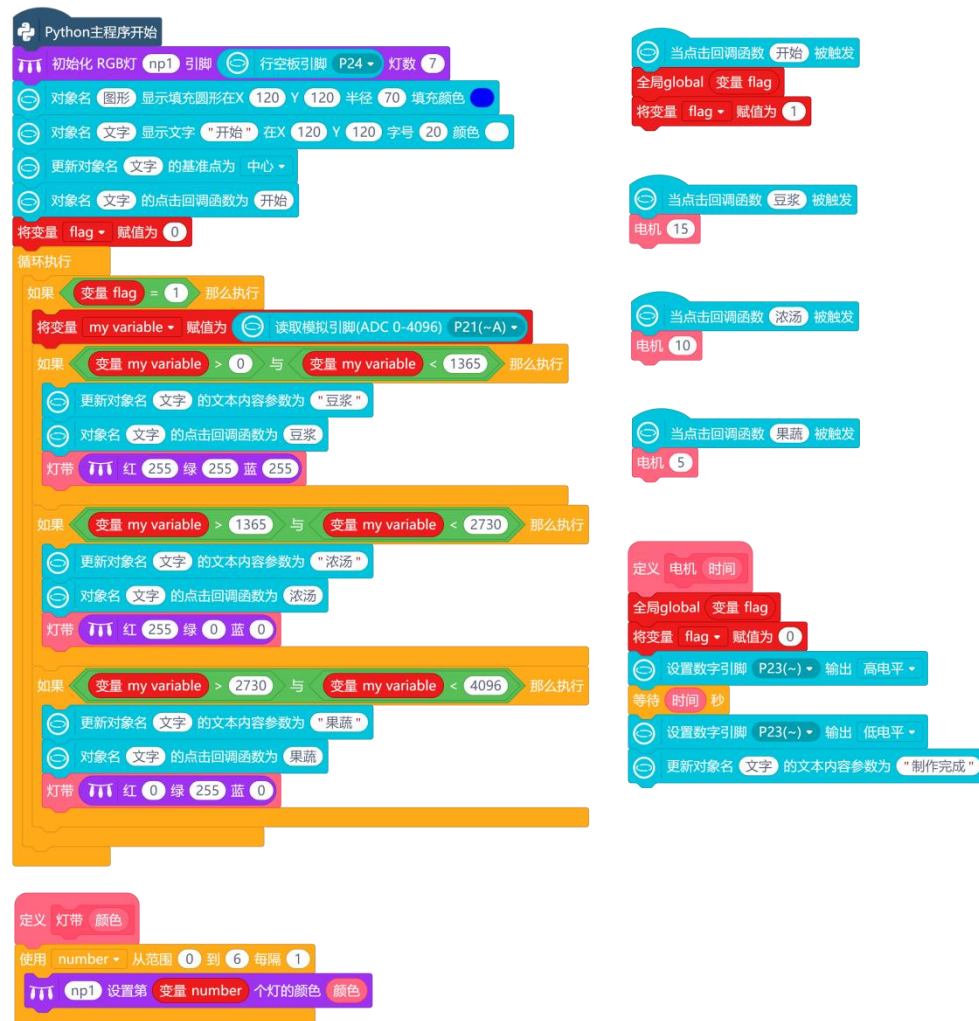
豆浆按钮被触发	
浓汤按钮被触发	



为了提高效率，我们还是将电机程序封装成一个函数名为“电机”，参数名称为“时间”的带参函数。



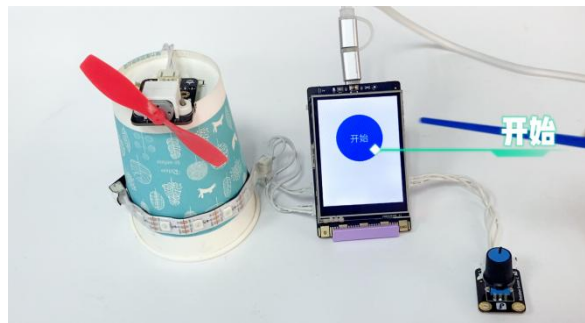
4. 参考程序



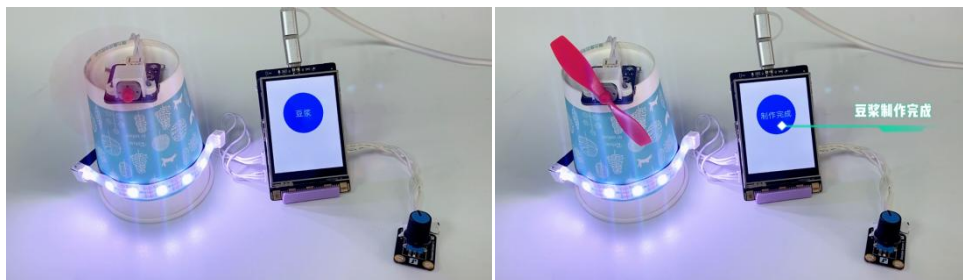
5. 程序运行

点击右上角的运行按钮，程序运行成功后，点击屏幕上的开始按钮进入菜单模式，然后旋转角度传感器选择菜单，菜单选择好之后点击菜单按钮启动料理机，等风扇停止后，表示该菜单已经制作完成。

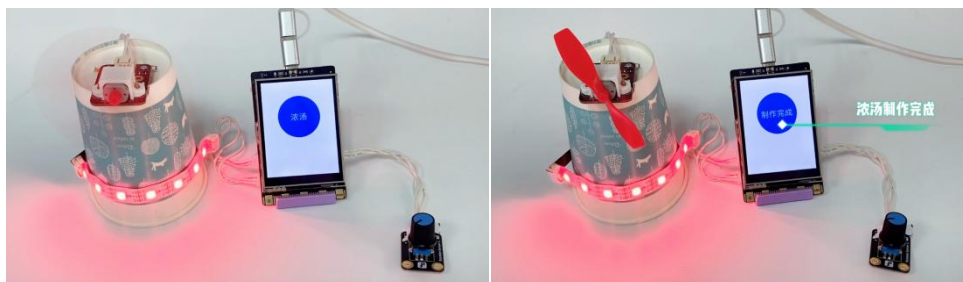
开始界面



豆浆模式



浓汤模式



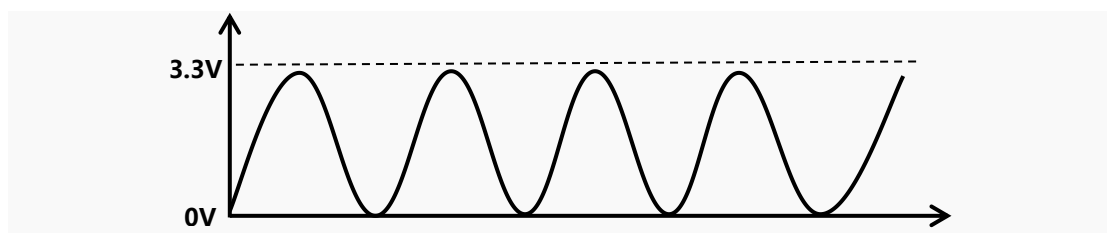
果蔬模式



知识园地

1. 认识模拟信号输入

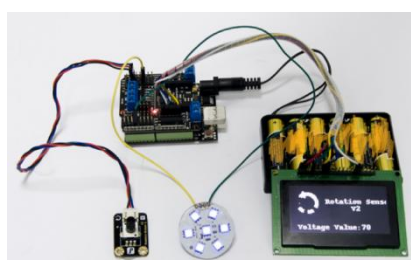
模拟信号是指用连续变化的物理量所表达的信息，如温度、湿度、压力、长度、电流、电压等等，我们通常又把模拟信号称为连续信号，它在一定的时间范围内可以有无限多个不同的取值。



模拟信号输入，就是模拟量传感器输出的 0-4096 的模拟值。将传感器连接在主控器的引脚上，利用 mind+ 中的“读取模拟引脚”的指令来读取它，这个过程就是模拟信号输入。

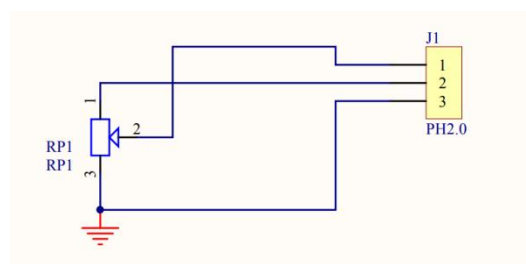
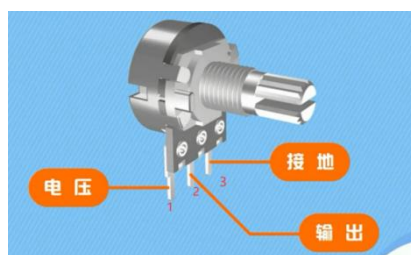
2. 认识角度传感器

这节课我们使用的输入模块是角度传感器，它是一款非常基本的模拟信号输入设备，与主控结合使用，可以轻松地完成与旋转位置相关的互动效果，比如用它调整输入电压，能够调整 LED 灯的亮度。



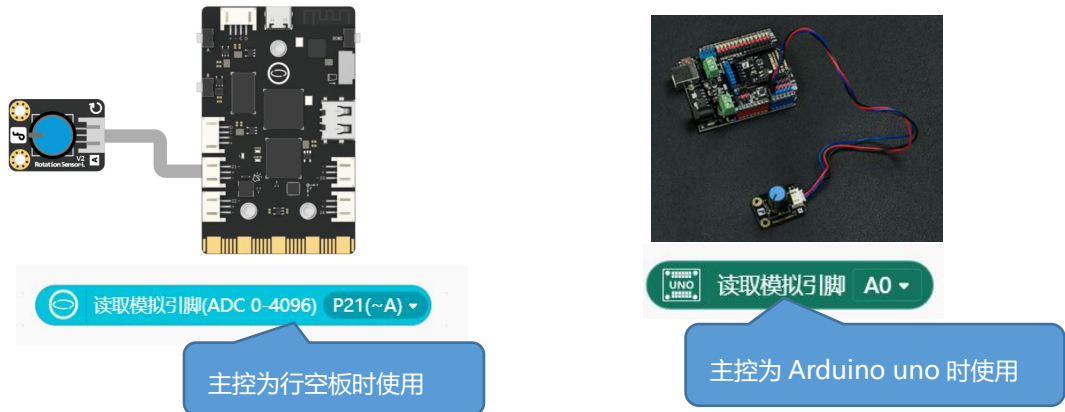
1) 角度传感器工作原理

角度传感器其实是基于电位器的旋转传感器。作为可变分压电阻用时，它的 3 个引脚是这样连接的，1 脚接输入电压，2 脚接输出，3 脚接地。当我们转动旋钮时，就是 2 脚在 1 与 3 脚间滑动来调节电阻大小，从而控制电压。角度传感器的旋转角度为 0~300 度，当旋转至 0 度时，传感器输出的模拟值为 0，当旋转至 300 度时，输出的模拟值为 4096。



2) 角度传感器的使用方法

角度传感器可以直接插在扩展板或者控制器的端口上使用，使用方法见下图，首先找一根 3pin 的连线，然后将整个模块连接到控制器或者扩展板上。然后在 mind+ 中，用读取模拟引脚的指令来读取它的值。在行空板的模拟口中，已经将 0V 到 3.3V 之间的值映射为 0-4096 范围内的模拟值。比如，0 对应 0V，4096 对应 3.3V。在 Arduino 的模拟口中，已经将 0V 到 3.3V 之间的值映射为 0-1024 范围内的模拟值。比如，0 对应 0V，1024 对应 3.3V。



3. 常用的 RGB 颜色对照表

	R	G	B		R	G	B
白色	255	255	255	青色	0	255	255
黑色	0	0	0	紫色	255	0	255
红色	255	0	0	黄色	255	255	0
绿色	0	255	0	橙色	255	97	0
蓝色	0	0	255	灰色	192	192	192

4. 指令学习

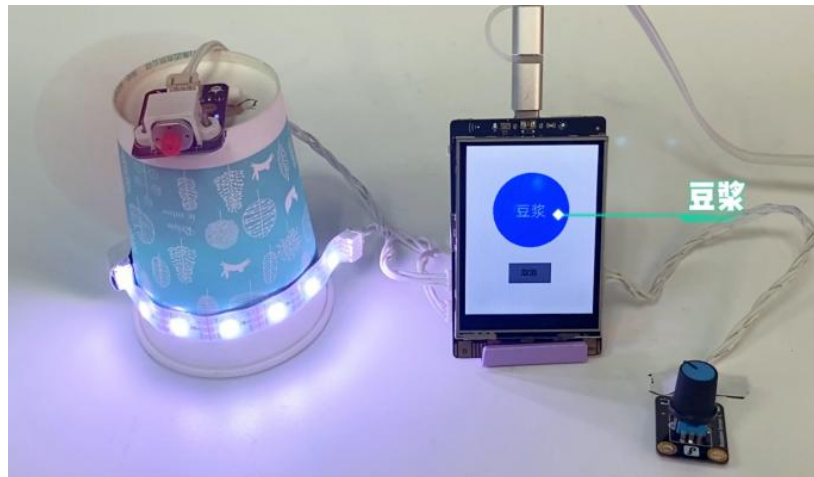
动手实践里主要使用了以下操作指令。

Block	功能说明
	该指令用于在行空板上显示填充的圆形，在指令中可以设置 X,Y 坐标、圆的半径以及填充颜色。
	该指令用于修改文字对象的文本内容。使用时，需写明要修改的对象名称和修改的文字。
	该指令用于读取模式引脚的输入值。
	该指令用于定义线程对象名启动后，要执行的操作。使用时，需写明停止的线程对象名。
	该指令用于生成一个线程对象。使用时，只需要放置在需要同时运行的程序部分，就可以启动该线程。
	比较运算符在代码中是用作判断的，用于对常量、变量或表达式的结果进行大小比较。比较运算符返回的结果只有两种，true 和 false。
	逻辑运算符“与”，只有当两个操作数的值都为真时，运算结果为 true (真)，否则为 false (假)。这个“与”就相当于日常生活中表达的“并且”。

挑战自我

在项目中，我们已经实现了料理机的基本功能。接下来，请完成两个挑战。

1. 增加取消按钮，当料理机正在工作或者工作完之后，点击取消按钮，可以回到开始界面，重新开始选择菜单。



2. 使用“对象名 显示仿数码管字体 () 在 x ()y() 字号() 颜色 蓝”指令修改任务三的程序，为每个菜单模式添加倒计时的功能。



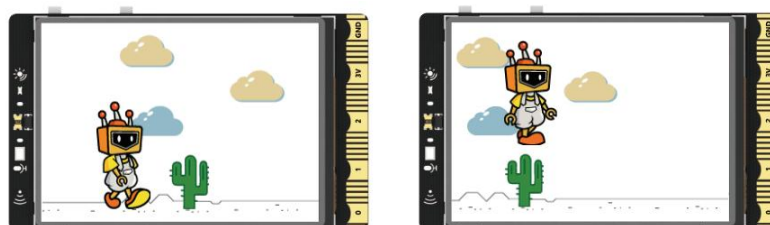
第 6 课 避障小游戏

谷歌浏览器里面隐藏了一个类似跑酷的避障小游戏。它的玩法很简单只要你轻轻敲击空格键，小恐龙便会跳起来躲避障碍物。现在是不是迫不及待地想试一试了！本节课，我们就来学习一下如何用行空板和按钮模块制作一款避障小游戏，让 mind+角色体验不一样的游戏旅程。



任务目标

利用 mind+角色、仙人掌、云朵背景布置游戏界面，然后利用按钮控制 mind+角色跳跃。例如：仙人掌和云朵背景会连续向左移动，当仙人掌要触碰到 mind+角色时，按下按钮，让 mind+角色跳起来躲避仙人掌。

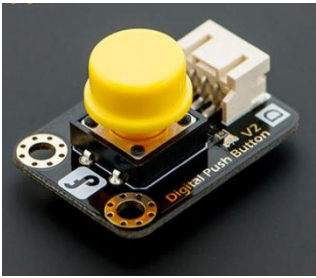


知识点

1. 掌握按钮模块的工作原理及使用方法。
2. 掌握图片连续移动的方法。
3. 利用坐标知识判断目标与障碍物的位置。
4. 掌握多线程的使用方法。

材料清单

硬件清单：

	
行空板 x1	按钮模块 x1
	
白色硅胶绞线 x1	Type-C USB 线 x1

软件使用：Mind+编程软件 x1

下载地址：<https://mindplus.cc/>

动手实践

避障小游戏，主要是模拟谷歌浏览器的小游戏恐龙快跑，通过按钮模块控制 mind+角色跳起来躲避障碍物。而且这次的界面与以往不同的地方是，背景与角色都会动起来。接下来，通过三个简单的小任务来学习如何制作一个避障小游戏。

任务一：让 mind+动起来

在此项任务中，将背景图片、障碍物图片与 mind+动作图片导入文件系统，并切换 mind+的动作图片，让 mind+动起来。

任务二：控制背景与障碍物向左移动

在完成 mind+动起来的功能后，我们将学习开启多线程的方法，同时让背景图片与障碍物图片连续向左移动。

任务三：控制 mind+躲避障碍物

在任务二的基础上，加上按钮模块，当按钮模块被按下后，控制 mind+角色向上跳跃。如果 mind+未成功躲避仙人掌，所有素材停止移动，屏幕上显示“game over”的信息。

任务一：让 mind+动起来

1. 硬件连接

使用 USB 线将行空板连接到计算机。



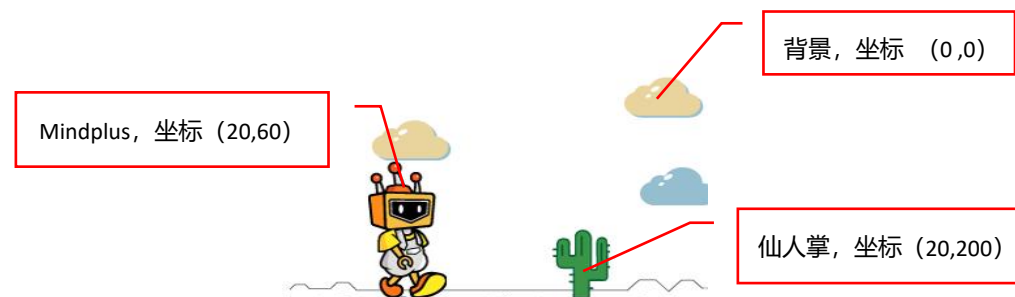
2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。

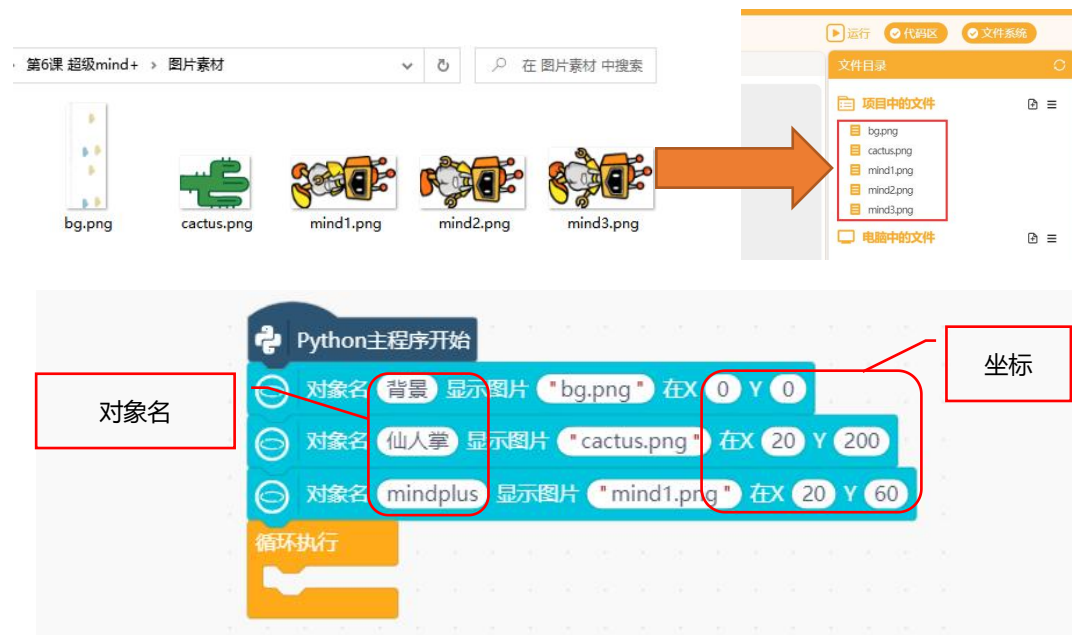


3. 编写程序

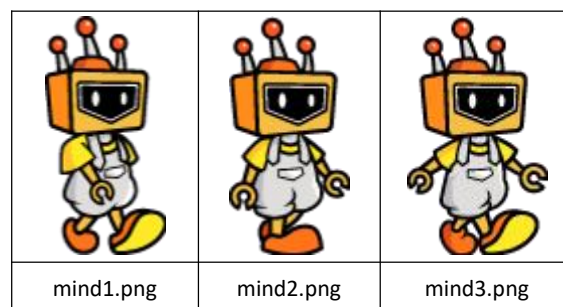
首先观察一下，任务一的界面上都包含哪些元素？一张背景图，一个 mind+角色与仙人掌图片，效果如下图。



- (1) 首先将上面提到的图片素材导入至文件系统。然后在“python 主程序”下用“显示图片”指令设置对象名及坐标。



- (2) 因为任务一的功能是让 mind+角色动起来，所以我们需要使用“更新图片”指令，更新对象名为“mindplus”的图片，通过让“mind1.png、mind2.png、mind3.png”这三个动作图片在屏幕上循环显示来制作 mind+动起来的动画效果。



4. 参考程序



5. 程序运行

检查行空板是否已连接到 mind+, 无误后, 点击右上角的运行按钮, 程序运行成功后, 行空板屏幕上会显示一个正在原地踏步的 mind+。



任务二：控制背景与障碍物向左移动

前面我们讲过, 任务二的功能就是让背景图片与障碍物图片连续向左移动。所以我们接下来的任务就是让“背景”与“仙人掌”不停地从右向左移动。



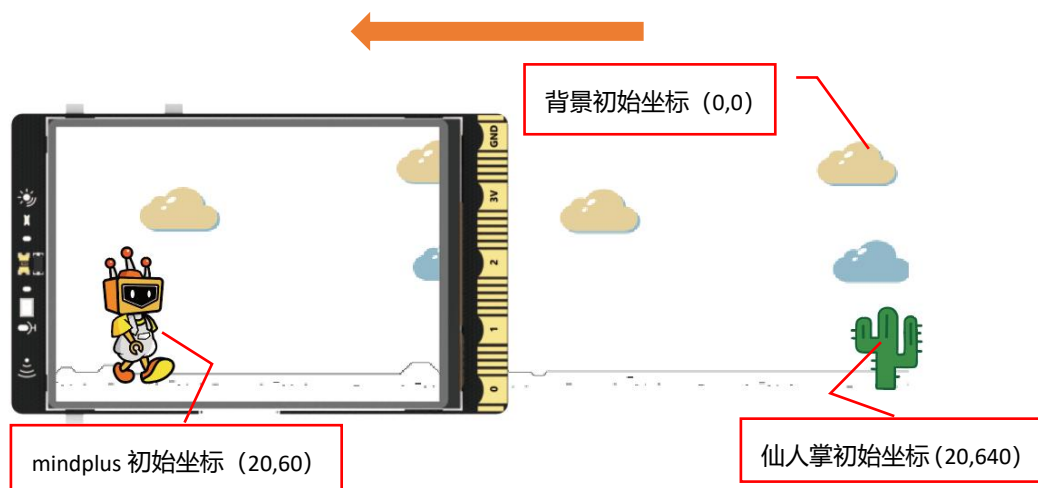
行空板屏幕正对自己, Type-C USB 接口处为左边, 金手指这边为右边。

1. 准备工作

按照任务一的“[硬件连接](#)”与“[软件准备](#)”连接好行空板并准备好 mind+软件。

2. 编写程序

在程序开始之前，先分析一下任务二需要完成哪些功能？首先 mind+角色要持续运动，仙人掌与云朵背景从屏幕右边向屏幕左边持续移动。背景在移动的时候，中间不能出现空白，仙人掌需要等几秒才移动到屏幕上。



- (1) 由上图得知，想让仙人掌等几秒才移动到屏幕上，在最开始的时候，需要将仙人掌的坐标设置到屏幕外。因为整个背景的尺寸为 (240*640)，mind+角色的坐标为 (20,60) 所以将仙人掌的初始坐标设置为 (20,640)。



- (2) 在背景与障碍物向左移动时，mind+角色也需要同时运动，我们可以使用“线程”来完成同时执行两种功能。在行空板“多线程”分类下找到“[线程对象 thread1 启动](#)”和“[当线程对象 thread1 启动后执行](#)”两指令，搭配使用。

注意：“线程”常被用来实现某段时间多个功能同时运行，关于“线程”的介绍，参见“[知识园地](#)”。



- (3) 了解了多线程的用法后，现在只需要在界面显示完成后，启动线程，并在启动线程后执行相应的功能即可。首先我们启动 mind+动起来的线程，将“线程对象 thread1 启动”和“当线程对象 thread1 启动后执行”两指令拖动到脚本区，并把对象名修改成“thread_mind”。



- (4) 对象名称修改完之后，将“启动线程”指令放在主程序的循环执行前，并在“线程启动后执行”指令下更新 mindplus 角色动作。



- (5) 接下来我们用同样的方法将“线程”指令拖动到脚本区，并把对象名修改成“thread_bg_cactus”，表示该线程的功能是控制背景与仙人掌连续移动的。想要背景和仙人掌连续向左移动，就需要连续改变 Y 的坐标值。首先新建两个变量“bg_y”与“cactus_y”，并将它们声明为全局变量，主要用来存储背景与仙人掌的 Y 坐标值，然后将这两个变量分别赋值为“0”与“640”。

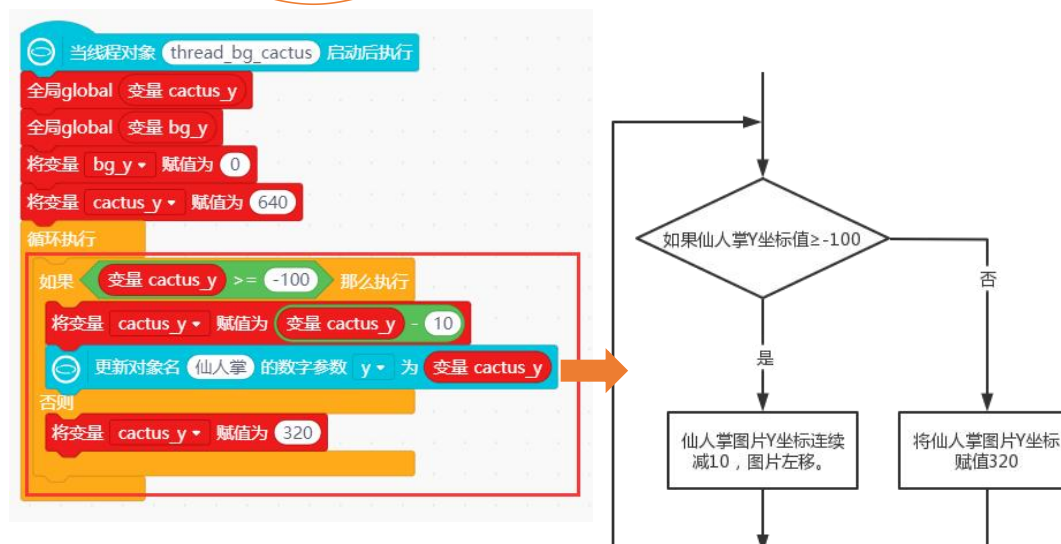
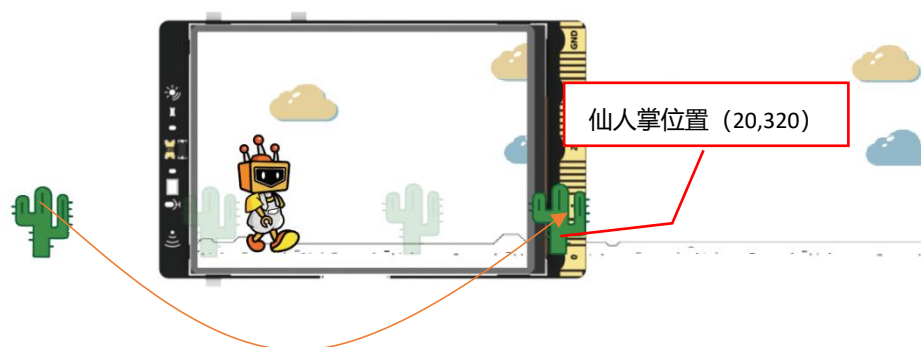


- (6) 现在我们已经确定了移动方向、背景与仙人掌的初始 Y 坐标。接下来就是控制图片移动。首先完成仙人掌从屏幕右边向左移动，具体分析如下：

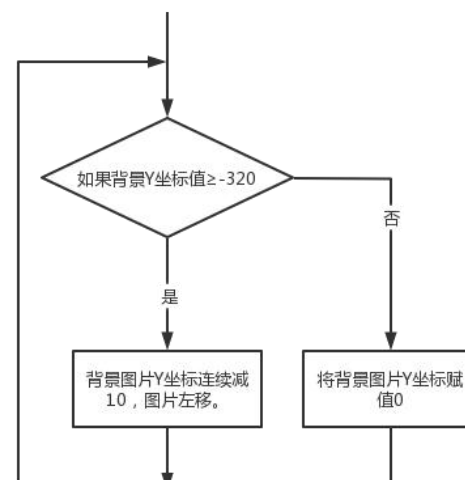
因为仙人掌想达到的显示效果是从右边进入屏幕，从左边出屏幕。所以初始和最终的坐标位置都应该超出屏幕 Y 坐标最大 (320) 和最小 (0) 的位置。设置初始位置为 640，是因为在游戏开始前，让仙人掌等几秒才出现在屏幕中，所以将初始坐标设置的远一点。



为了在游戏后期保持一个好的节奏，当仙人掌移动到最左端的坐标后，就将下一次最右边的 Y 坐标更新为 320。



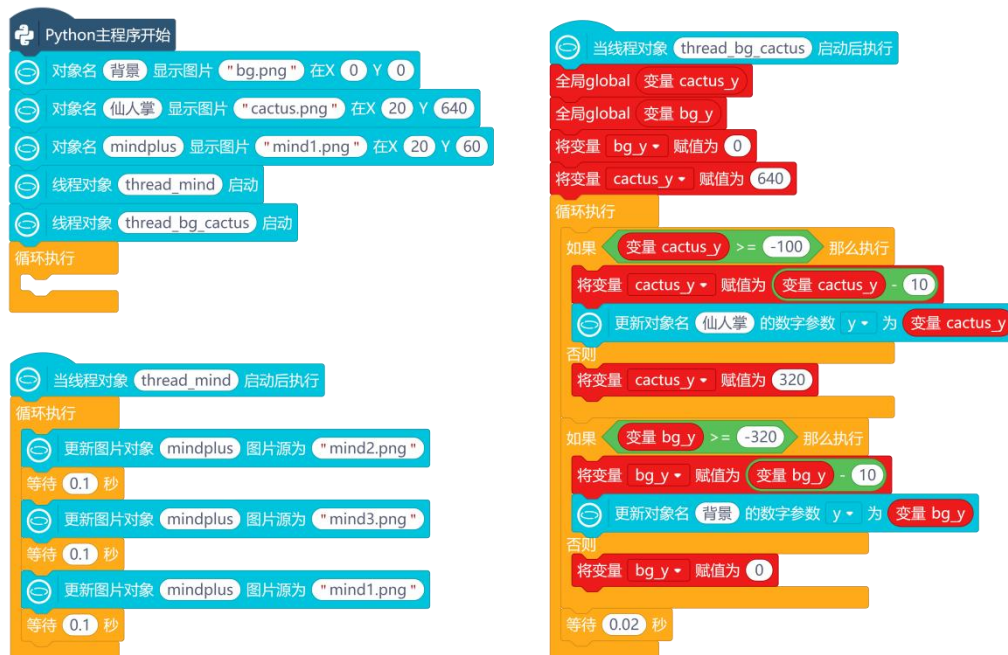
- (7) 完成了仙人掌向左移动的功能，接下来控制背景图片向左移动，首先我们确认图片在屏幕中的尺寸 X:240, Y:640，然后根据 Y 坐标长度找到背景图片合适的最左端坐标。根据测试，背景图片最左端的 Y 坐标应该设置成 -320 效果最佳。



- (8) 完成了控制背景与仙人掌连续移动的子线程功能后，将“线程对象 thread_bg_cactus 启动”指令放在主程序的循环执行前。



3. 参考程序



4. 程序运行

点击右上角的运行按钮，程序运行成功后，你会看见，背景与仙人掌都在向左移动。



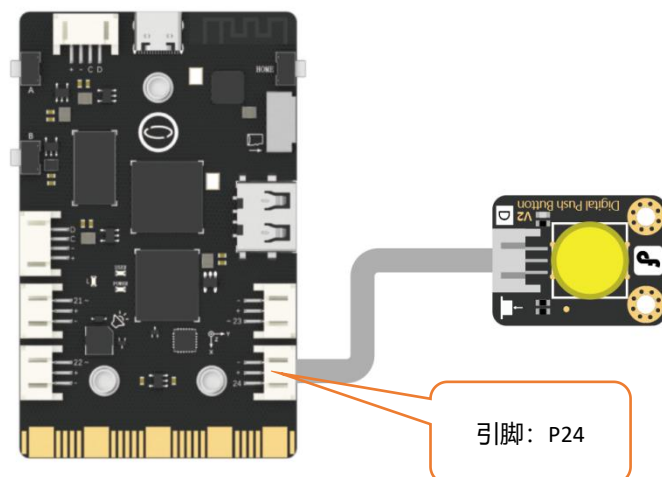
任务三：控制 mind+ 躲避障碍物

通过任务二，我们已经完成了让背景与仙人掌连续向左移动，接下来，我们将接入传感器——按钮模块，控制 mind+跳起来躲避仙人掌。

1. 硬件连接

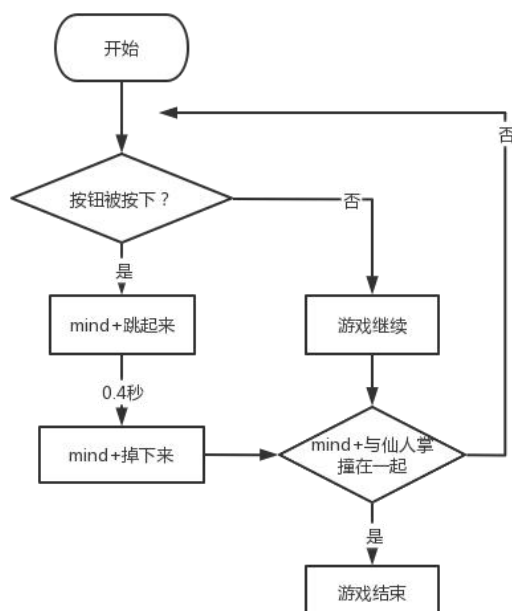
使用连接线将按钮模块连接到行空板上的 P24 脚，如下图。硬件连接成功后，使用 USB 线将行空板连接到计算机。最后打开 Mind+，按照任务一完成软件准备过程。

注意：关于“按钮模块”的相关介绍，见知识园地。



2. 编写程序

在编写程序前，先梳理一下任务三需要完成什么功能？在任务二的基础上，加上按钮模块，当按钮模块被按下后，控制 mind+角色向上跳跃，如果 mind+没有成功躲避仙人掌，所有素材停止移动，屏幕上显示“game over”的信息；反之，所有的素材则继续运动，游戏继续。

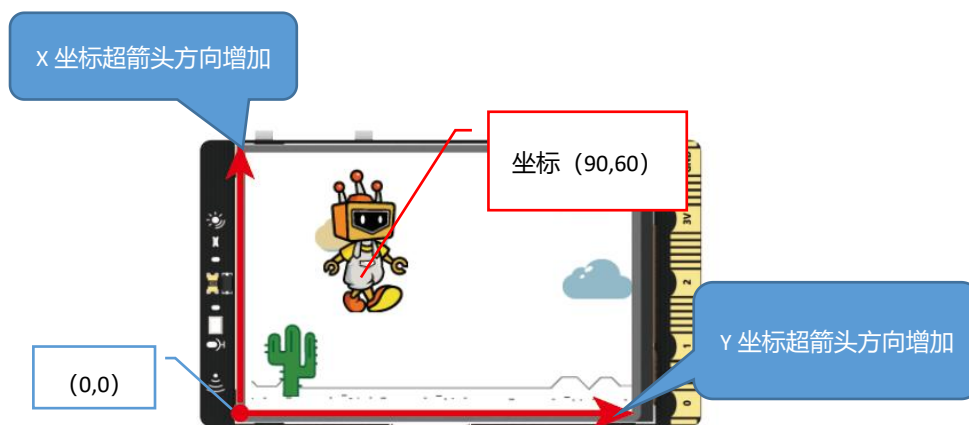


- 由流程图得知，当按下按钮后，mind+角色会先跳起来，0.4 秒过后掉下去。为了优化游戏体验，我们将使用标志变量 **flag** 来触发 mind+跳起来的功能。首先我们在最开始将 flag 赋值为 0，然后去主程序里的循环执行中判断它，当 **flag=0** 时，就判断按钮是否被按下，如果按钮被按下，就将变量 **flag** 的值

赋值为 1。

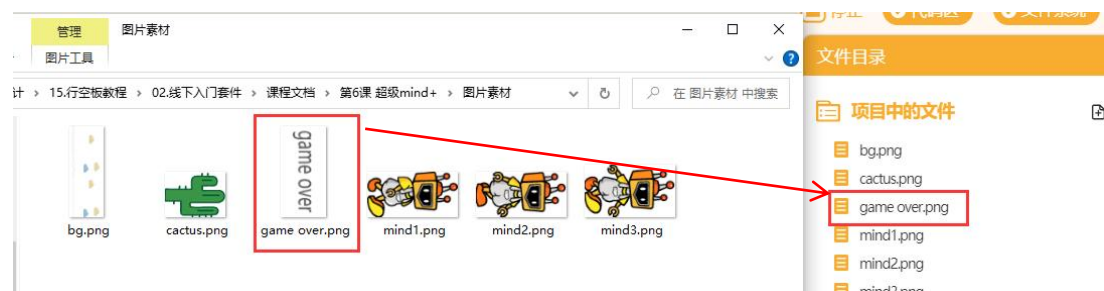


- (2) 现在已经知道，当按钮被按下后，flag 变量的值就被赋值为 1，那么接下来，我们就判断当 flag=1 时，mind+就跳起来。其实想让 mind+跳起来很简单，只需要保持 Y 坐标不变 (Y=60)，改变 mind+的 X 坐标就行。根据测试，mind+跳起来的位置为 (90,60)，掉下来的位置为 (20,60) 比较合适。



- (3) 到目前为止，按下按钮，mind+已经会向上跳跃了，接下来我们就来实现，mind+与仙人掌相撞后，终

止游戏的功能。首先在图片素材文件夹中将图片“game over.png”导入至文件系统。



- (4) 因为在游戏中运行时，“game over”的图片不能显示在屏幕上，所以需要在主程序下，将它设置在屏幕外，当 mind+与仙人掌相撞后再更新坐标将它移动到屏幕上即可。

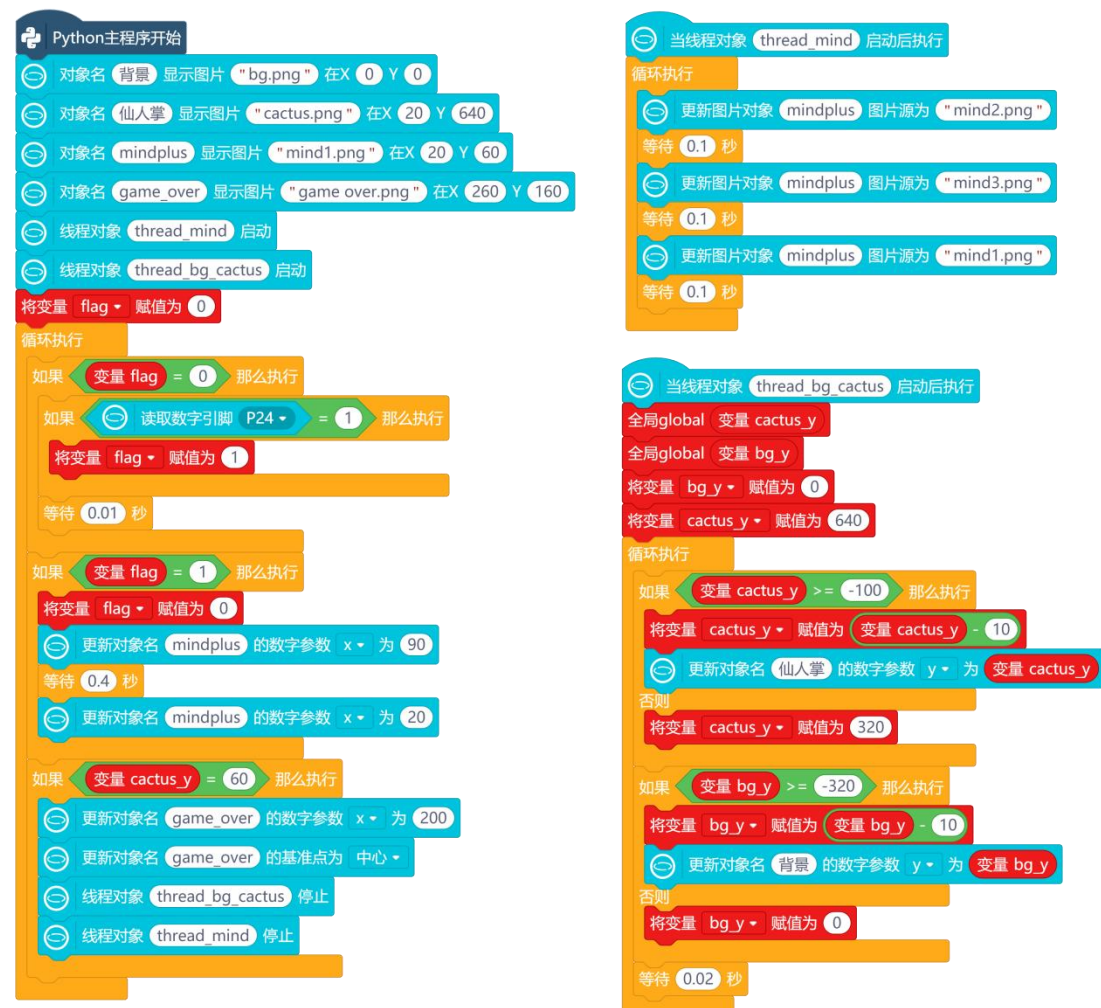


- (5) 由流程图得知，当 mind+与仙人掌相撞后，就表示游戏结束，这时屏幕上会显示“game over”的信息，并终止所有的线程。其实想要确定 mind+角色是否与仙人掌相撞，只需要判断它俩的 Y 坐标是否重合。





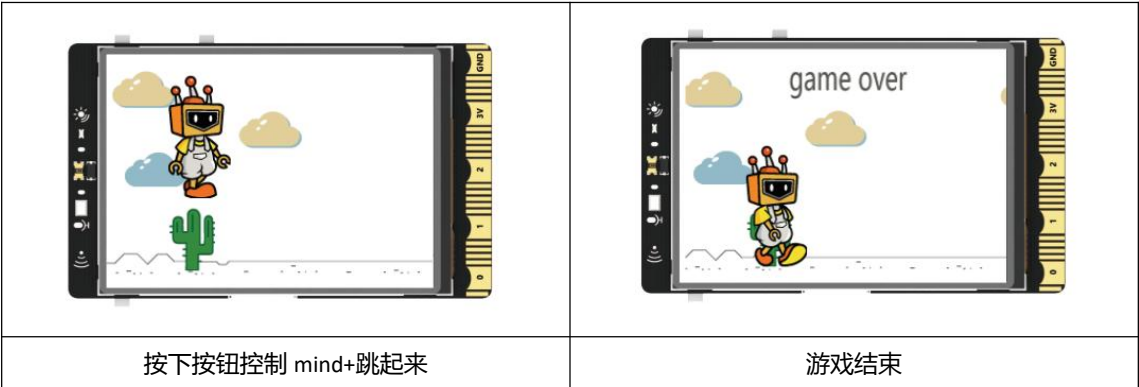
3. 参考程序



4. 程序运行

点击右上角的运行按钮，程序运行成功后。当 mind+角色与仙人掌接近时，按下按钮，让 mind+越过仙人掌，越过仙人掌后，mind+继续向前进；如果 mind+没有越过仙人掌，屏幕上则显示“game over”的

信息。如果想要继续下一次游戏，需要重新运行程序。



5. 试一试

通过上面的程序，你是否发现当游戏结束后，按下按钮，mind+角色还是会向上跳跃？那么请试着修改程序，当游戏结束后，按下按钮 mind+角色还是停止不动。

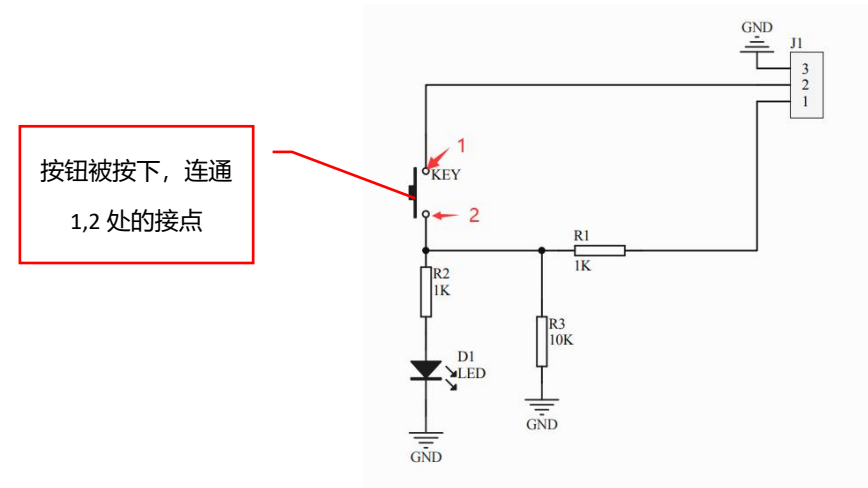
知识园地

1. 认识按钮模块

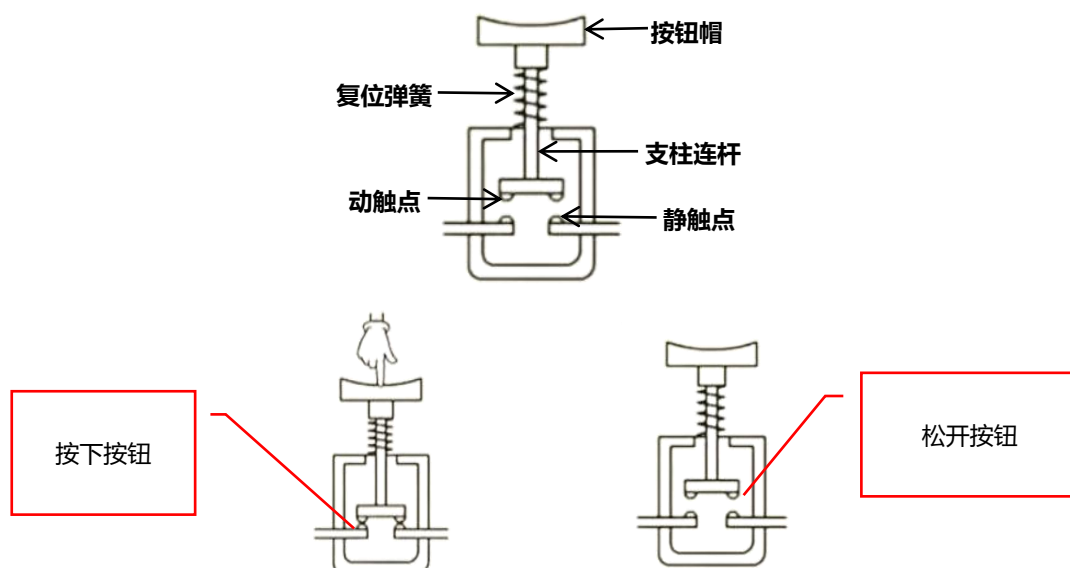
这节课，我们认识了一个新的数字输入模块——按钮，它是一种手动控制器。因为按钮只能短间接通或者断开电流，因此按钮常在控制电路当中发布操作指令，接通或者断开电路，用来控制机械与电气设备的运行。当它与主控结合时，可以完成许多有趣的互动效果。

1) 按钮模块工作原理

按钮模块的工作原理是通过按下按钮来使电路发生变化，从而实现控制电路的开关状态。当按钮被按下时，按钮模块内部的接点就会闭合，从而使电路闭合。当按钮被松开时，按钮模块内部的接点就会断开，从而使电路断开。

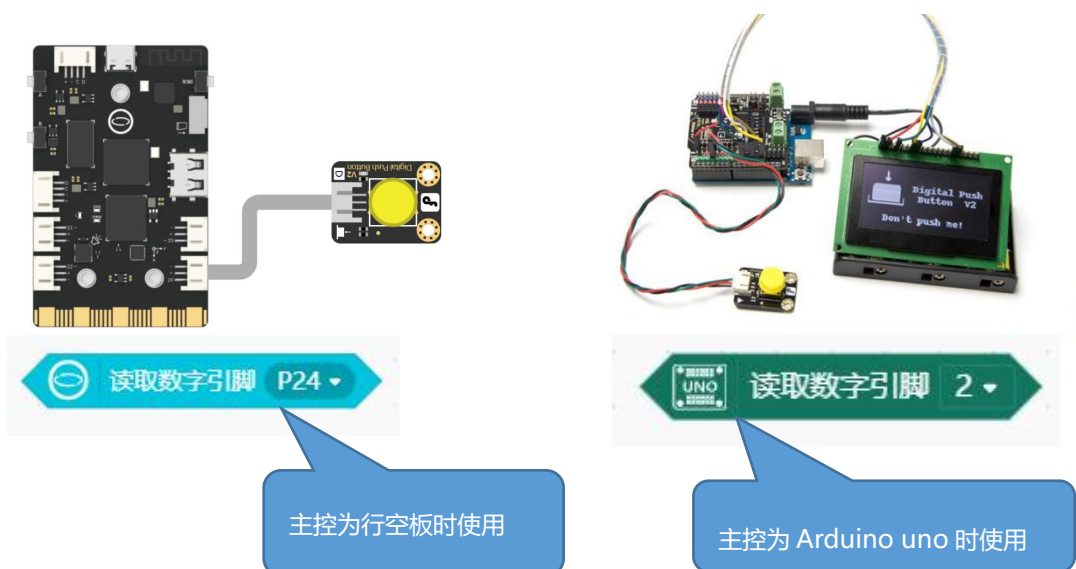


按钮模块通常由按钮帽、复位弹簧、支柱连杆、动触点、静触点以及外壳等组成。当按钮被按下时，支柱连杆会向下移动，从而使弹簧收缩。同时，动触点与静触点之间的距离也会缩短，最终导致接点闭合。当按钮被松开时，弹簧会弹回，支柱连杆返回到原位，同时按钮与接点之间的距离也会增加，最终导致接点断开。



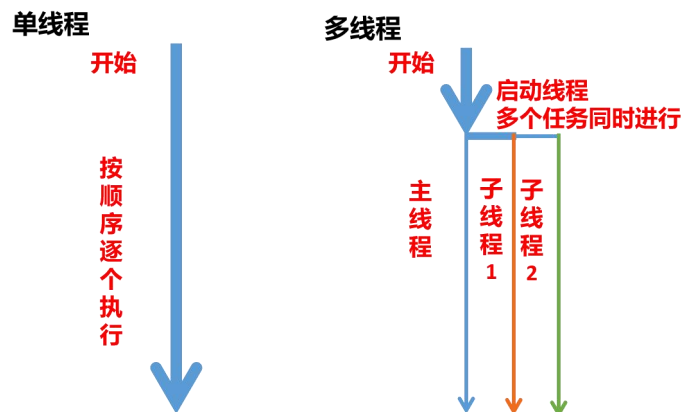
2) 角度传感器的使用方法

角度传感器可以直接插在扩展板或者控制器的端口上使用，使用方法见下图，首先找一根 3pin 的连线，然后将整个模块连接到控制器或者扩展板上。然后在 mind+ 中，用读取数字引脚的指令来读取它的值。



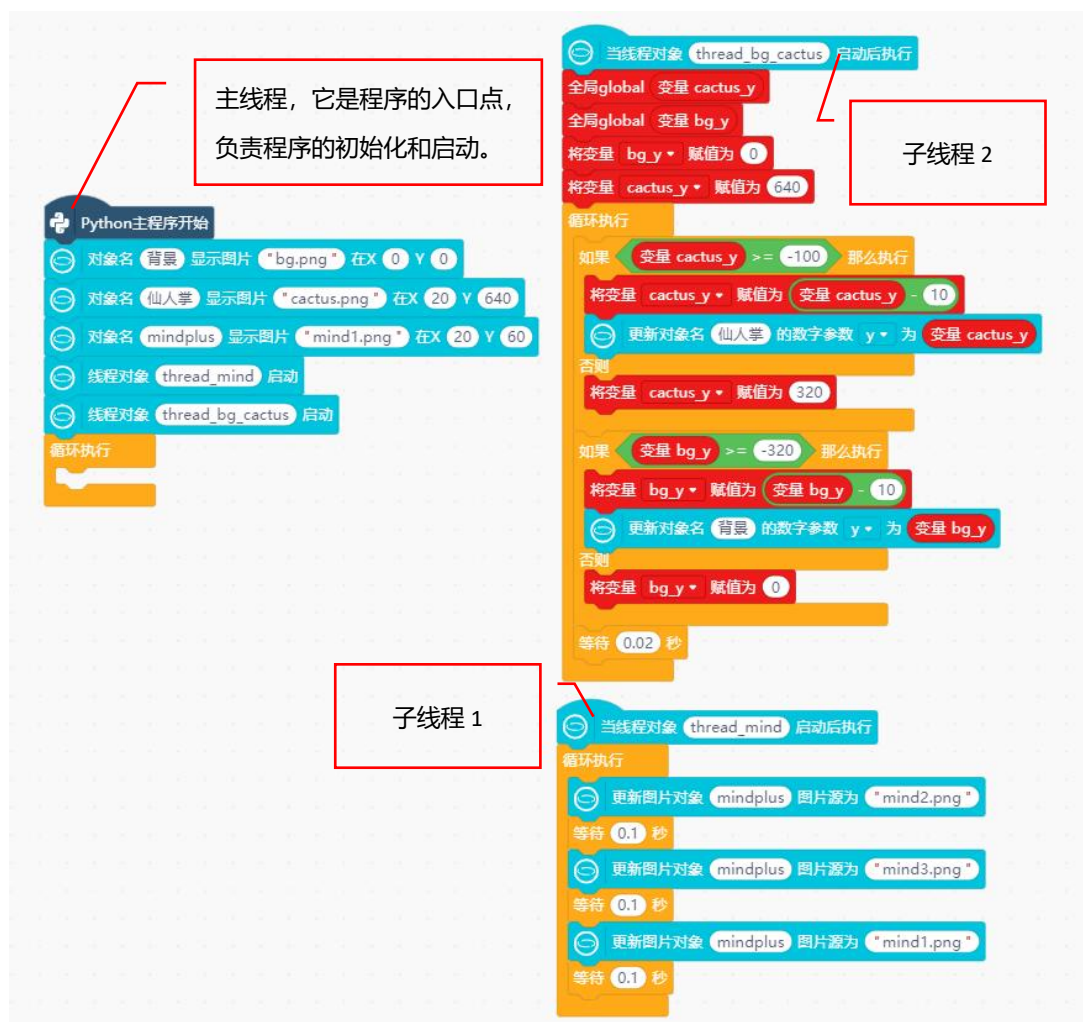
2. 程序中的线程

线程，它是一个程序在运行过程中正在进行的任务，需要说明的是这里的任务是指一系列有先后执行顺序的代码，你可以把它类比为“线”。当一个程序只有一个任务时，通常我们说这个程序是单线程的，例如任务一；但是当一个程序中需要同时执行多个任务时，这个程序就需要使用多个线程，来同时完成多个子任务，我们称从开始到最后一直执行的线程为主线程，其他子任务为子线程。



你可以类比在家做饭帮助理解单线程和多线程，当只有一个灶头时，你需要先完成蒸鸡蛋，然后再炒菜，两个菜只能按顺序制作，效率比较低，但当有两个灶头时，你可以在蒸鸡蛋的同时完成炒菜任务，这样效率就提起来了。

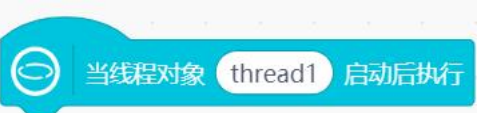
在这个项目中，任务二和任务三就是多线程案例。



另外，由于线程启动和运行会分掉主线程时间，线程使用越多，程序执行就越慢，甚至导致程序卡死，所以建议谨慎使用。

3. 指令学习

动手实践里主要使用了以下操作指令。

Block	功能说明
	该指令用于生成一个线程对象。使用时，只需要放置在需要同时运行的程序部分，就可以启动该线程。
	该指令用于停止指定的线程对象。使用时，需写明停止的线程对象名。
	该指令用于定义线程对象名启动后，要执行的操作。使用时，需写明停止的线程对象名。
	读取数字引脚的输入值。
	大于等于比较运算符，用于对常量、变量或表达式的结果进行大小比较。比较运算符返回的结果只有两种，true 和 false。

挑战自我

在项目中，我们已经完成了一个避障小游戏，但是每次游戏结束后，需要重新运行程序才能再次开始游戏。接下来，请添加一个刷新按钮，当游戏结束后，点击刷新按钮重新开始游戏。



第 7 课 AI 语音管家

很小的时候，我有一个梦想，梦想着自己拥有魔法，可以通过语音“芝麻开灯”“芝麻关灯”，来控制灯光的开启与关闭。随着智能家电的普及，语音控制也逐渐走入我们的生活，使用语音指令控制灯光就不再是梦想了。



接下来，就使用行空板搭配一些硬件模块，制作一款 AI 语音管家。要控制家里的设备，就是一句话的事儿~

任务目标

通过语音的方式，控制灯、风扇、窗户。例如，长按行空板上的 A 按钮，对着麦克风说：“打开灯”，RGB 灯带亮；按住行空板的 A 按钮，对着麦克风说：“关闭灯”，RGB 灯带灭。

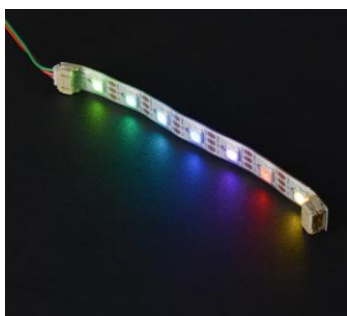
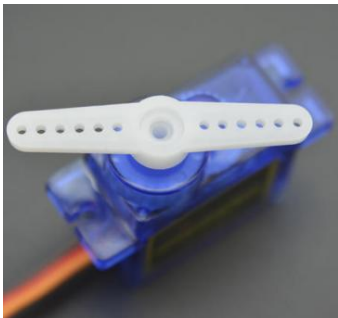


知识点

1. 了解语音识别技术。
2. 掌握语音指令控制硬件的使用方法。
3. 了解舵机的工作原理及使用方法。

材料清单

硬件清单:

	
行空板 x1	RGB 全彩灯带 x1
	
舵机 x1	风扇模块 x1
	
白色硅胶绞线 x2	Type-C USB 线 x1

软件使用: Mind+编程软件 x1

下载地址: <https://mindplus.cc/>

动手实践

AI 语音管家的主要功能是让连接在行空板上的舵机、风扇、灯带能够听懂我们语音指令，然后根据指令完成对应的操作。接下来，就从下面三个任务来完成吧！

任务一：设计 AI 语音管家的控制界面

在此项任务中，利用图片素材完成 AI 语音管家的控制界面设计。然后当按钮 A 被按下，更新麦克风图片与提示文本。

任务二：语音转文字

长按 A 按钮，录入的语音控制指令；松开 A 按钮，将语音指令转换为文字，并将文字显示在行空板上。

任务一：设计 AI 语音管家的控制界面

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。



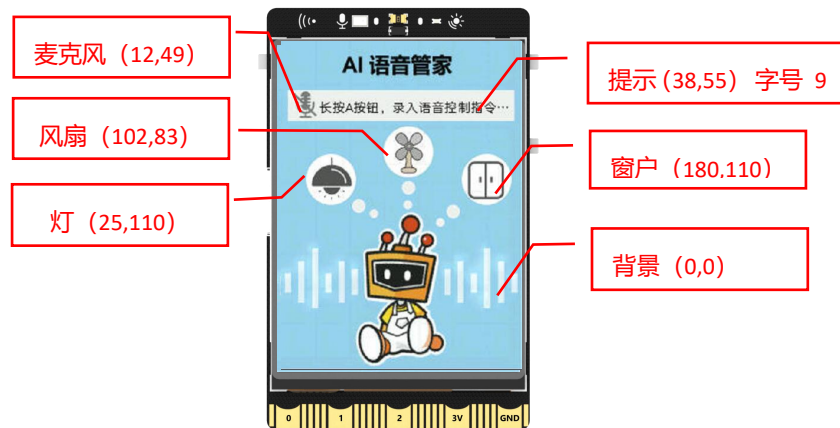
The screenshot shows the Mind+ software interface with several red boxes and arrows indicating the setup process:

- 1. 切换至 python 模式**: Points to the 'Python模式' button in the top right corner.
- 2. 切换到图形化编程模式**: Points to the '图形化' button in the top right corner.
- 3. 出现预设代码**: Points to the 'Python主程序开始' button in the center of the workspace.
- 4. 行空板库加载成功**: Points to the '行空板' button in the left sidebar.
- 5. 行空板连接成功 显示行空板 IP**: Points to the '行空板' button in the left sidebar.

The interface also shows a code editor on the right with a Python script and a terminal window at the bottom right displaying the connection status and IP address.

3. 编写程序

在开始编程之前，分析一下任务一需要实现的功能。首先在行空板上显示语音管家的控制界面，然后在录音区显示录音图标以及提示文字，最后再将灯、风扇、窗户的状态图显示在界面上。AI 语音管家控制界面用到的对象及对应坐标，如下图所示。



(1) 设计控制界面，先将图片素材库文件夹中的所有图片，导入到文件系统中。



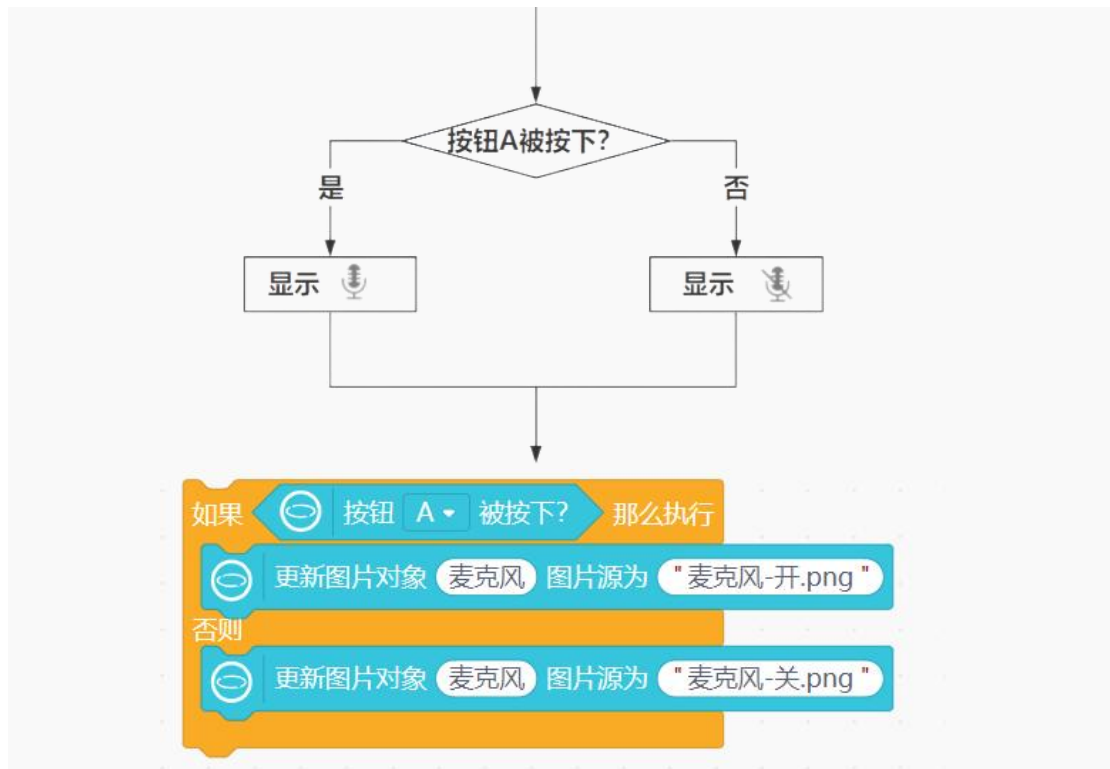
(2) 使用**显示图片**指令，将文件系统中的“背景.png”、“麦克风-关.png”、“风扇-关.png”、“灯-关.png”、“窗户-关.png”这些图片设置对应的XY坐标，显示在行空上。



- (3) 使用**显示文字**指令，将文字提示“长按 A 按钮，录入语音控制指令...”设置对应的 XY 坐标以及字号，显示在行空板上。

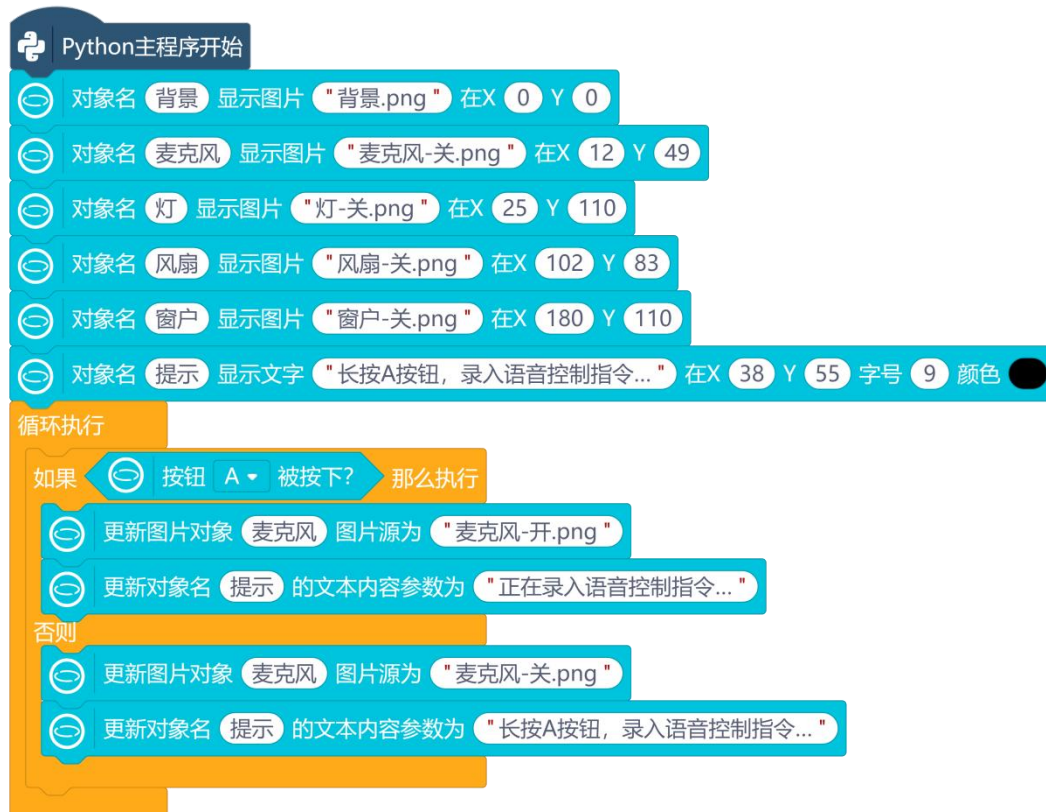


- (4) 接下来，在**循环控制**指令中，使用“**如果...那么执行...否则...**”指令，判断**按钮 A 是否被按下**。判断结果为真时，使用**更新图片对象**指令，更新图片源为“麦克风-开.png”；否则，更新图片源为“麦克风-关.png”。



当然在判断结果为真时，还可以使用**更新文本内容参数**指令，将提示对象中文字更新为“正在录入语音控制指令...”。

4. 参考程序



5. 程序运行

检查行空板是否已连接到本地地址 10.1.2.3，检查无误后，点击右上角的运行按钮。程序运行成功后，长按 A 按钮，麦克风图标更新为“启用”状态，提示文字更新为“正在录入语音控制指令...”；松开 A 按钮，麦克风图标更新为“禁用”状态，提示文字更新为“长按 A 按钮，录入语音控制指令”。



任务二：语音转文字

语音转文字的方法很简单，按住行空板的按钮 A 后，使用麦克风进行音频的录制，然后将语音指令转换成文字显示在行空板上。这个任务直接在任务一程序的基础上，添加语音转文字的功能即可。

1. 准备工作

- (1) 行空板连接无线网络，双击打开浏览器，在浏览器地址栏录入 10.1.2.3，进入行空板服务页面连接无线网络（若没有无线网可以使用手机热点）。

注意：连接无线网络时，仅支持 2.4GWiFi 热点。



- (2) 获取“讯飞开放平台”账户密钥，获取账号、密钥等信息，需要进入“讯飞开放平台”注册、申请应用服务。首先，使用浏览器访问“讯飞开放平台”网站：<https://www.xfyun.cn/>；然后，在首页右上角点击注册，进入注册界面。

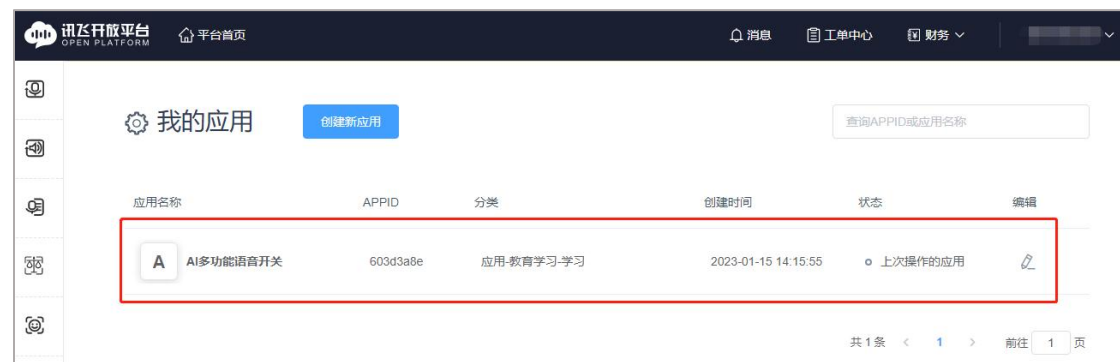


完成注册后，登录讯飞开放平台。点击右上角的“控制台”，进入应用界面，点击“创建新应用”，开始填写应用信息。

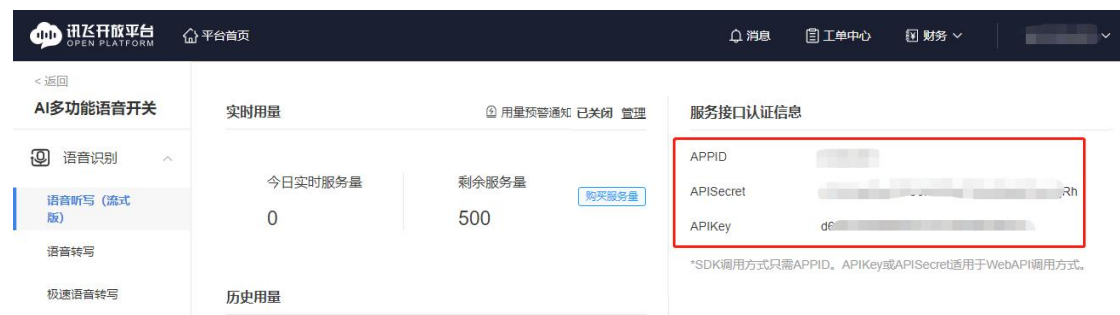


信息填写完成后，点击“提交”，完成应用的创建。现在就可以在“我的应用”中查看刚才创建成功

的应用。



点击应用，查看应用的服务账号、密码、密钥等信息。



- (3) 添加讯飞语音扩展库，首次使用用户库时，需要先手动添加网址，若直接打开已添加好库的程序，不用网址搜索添加，运行程序即可。添加方法如下：点击“扩展”，在“用户库”中录入网址：
<https://gitee.com/liliang9693/ext-xunfeiyuyin> 检索。



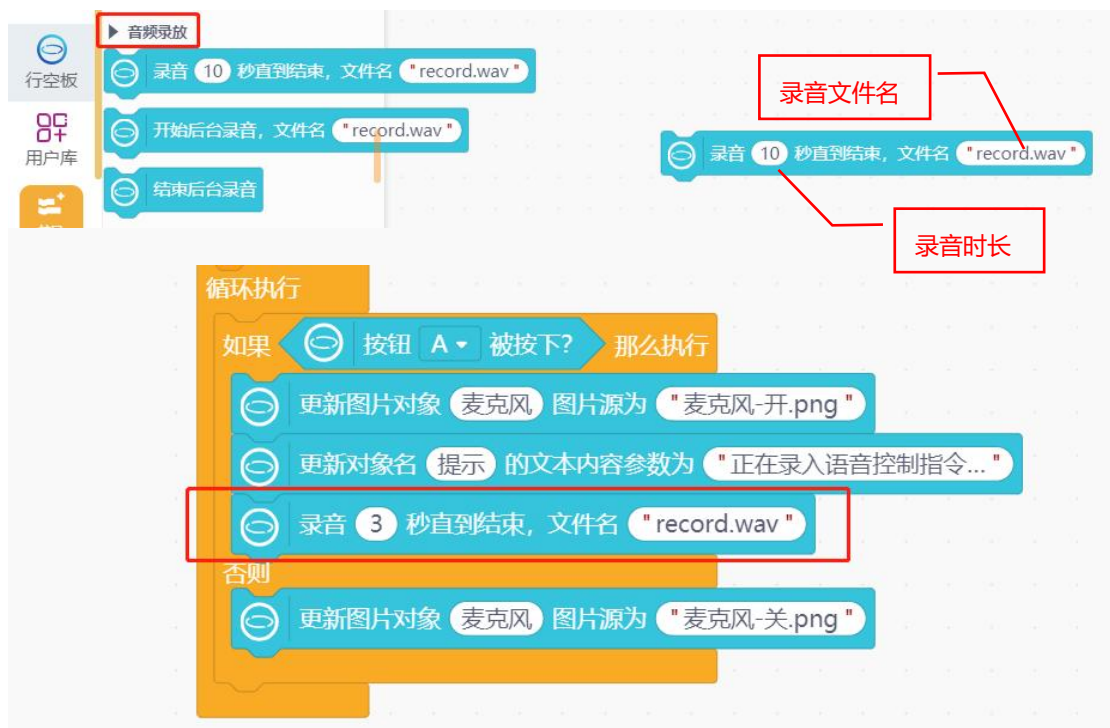
2. 编写程序

- (1) 使用讯飞语音的语音识别功能，对录制的语音进行识别。首先需要在 **Python 主程序开始** 下，使用 **讯飞语音设置账户** 指令，填写对应的服务账号、密码以及密钥信息。

注意：需要正确填写自己申请成功的账号、密码、密钥信息。



- (2) 在语音转换文字之前，需要先录入语音控制指令。行空板自带录音功能，你可以在“行空板”里的“音频录放”分类下找到 **录音 10 秒直到结束，文件名“record.wav”** 指令。使用时，设置好录音时长以及录音文件名即可。



- (3) 然后，使用 **讯飞语音 识别录音文件** 指令，对录入的语音指令进行识别。为了方便使用，新建一个 **变量识别内容**，将识别录音成功的文字赋值给 **变量识别内容**。最后，使用 **更新文本内容参数** 指令，更新提示对象的文本内容为 **变量识别内容**。



3. 参考程序



4. 程序运行

点击右上角的运行按钮，程序运行成功后，长按 A 按钮，进行语音指令录入，指令录完后，松开 A 按钮后，语音会被转换为文字在显示在行空板上，效果如下。



注意：项目运行时，可能会出现报错，可以尝试重新运行或断开行空板连接。出现以下报错时，请根据对应的方式进行修改。

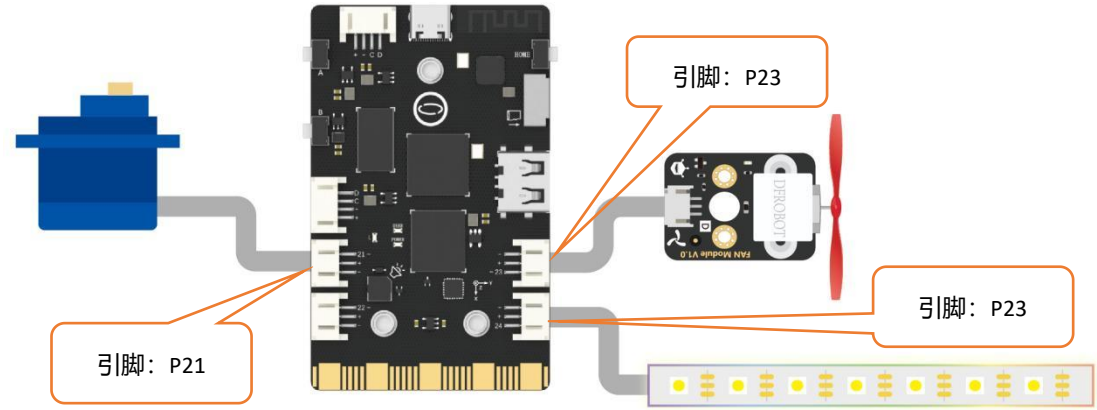
错误信息	修改方式
error: Handshake status 401 Unauthorized	当前行空板没有正确访问开放平台，此时需要更换无线网络并等待一会，再重试。
error: [Error -3] Temporary failure in name resolution	无线网络连接有问题，需要检查或更换无线网络。

任务三：制作语音控制系统

1. 硬件连接

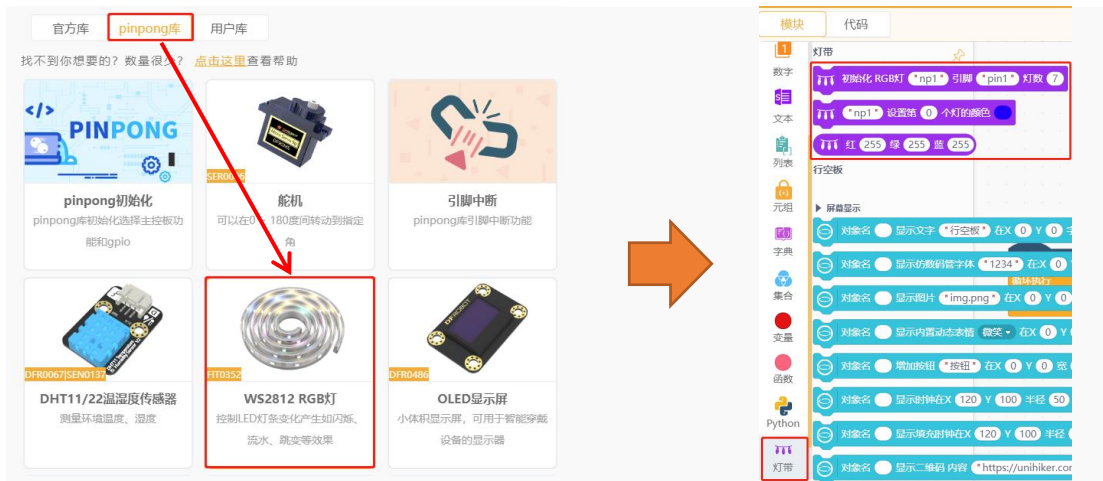
使用连接线将风扇模块连接到行空板的 **P23** 脚，将 RGB 灯带连接到行空板的 **P24** 脚，将舵机连接到行空板的 **P21** 脚，如下图。

注意：[关于“舵机”的相关介绍，见知识园地。](#)



2. 软件准备

- (1) 这个任务在任务二的基础上，进行程序的添加。
- (2) 添加 WS2812 RGB 灯库，点击“扩展”，在 pinpong 库中找到“WS2812 RGB 灯”并点击，完成添加。



(3) 添加舵机库，点击“扩展”，在 pinpong 库中找到“舵机”并点击，完成添加。



3. 编写程序

使用语音指令控制硬件，要知道有哪些语音控制指令，并且了解每条语音控制指令对应什么功能。语音控制指令的功能对应表如下：

语音控制指令	实现功能
打开灯	控制 RGB 灯带亮
关闭灯	控制 RGB 灯带灭
打开风扇	控制风扇开启
关闭风扇	控制风扇关闭
打开窗户	控制舵机转到 100 度
关闭窗口	控制舵机转到 10 度

- (1) 在 **Python 主程序开始** 之下，使用 **舵机初始化引脚** 指令，初始化舵机引脚为 **行空板引脚 P21**，然后使用 **舵机转动** 指令设置舵机转动角度为 10 度（窗户关闭）。

注意：在任务二程序的基础上，进行添加。



- (2) 在 **Python 主程序开始** 之下，使用 **RGB 灯初始化** 指令，初始化 RGB 灯引脚为 **行空板引脚 P24**，然后设置灯数为 7。



- (3) 新建 **函数语音控制功能**，在语音指令识别成功后，**调用函数语音控制功能** 即可。语音指令控制硬件的功能，在函数下实现。



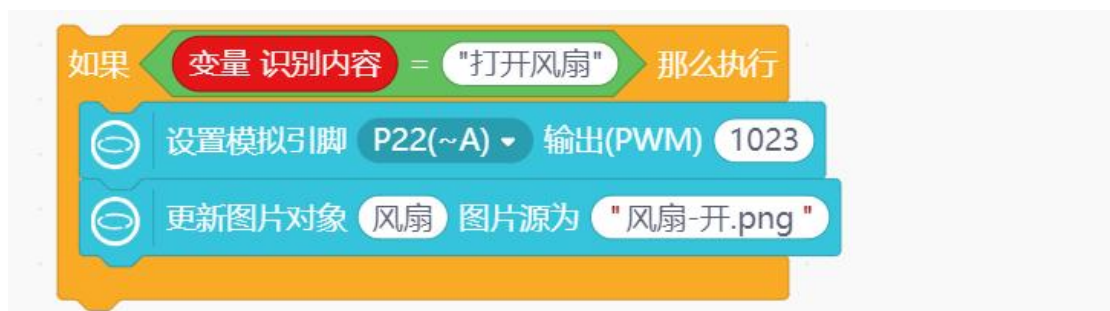
- (4) 在 **语音控制函数** 下，使用 **条件** 指令判断 **变量识别内容** 是否为语音控制指令中的内容。例如，当变量识别内容为“打开灯”时，使用 **设置灯带颜色** 指令，控制 RGB 灯带亮。然后使用 **更新图片对象** 指令，更新灯对象的图片源为“灯-开.png”。



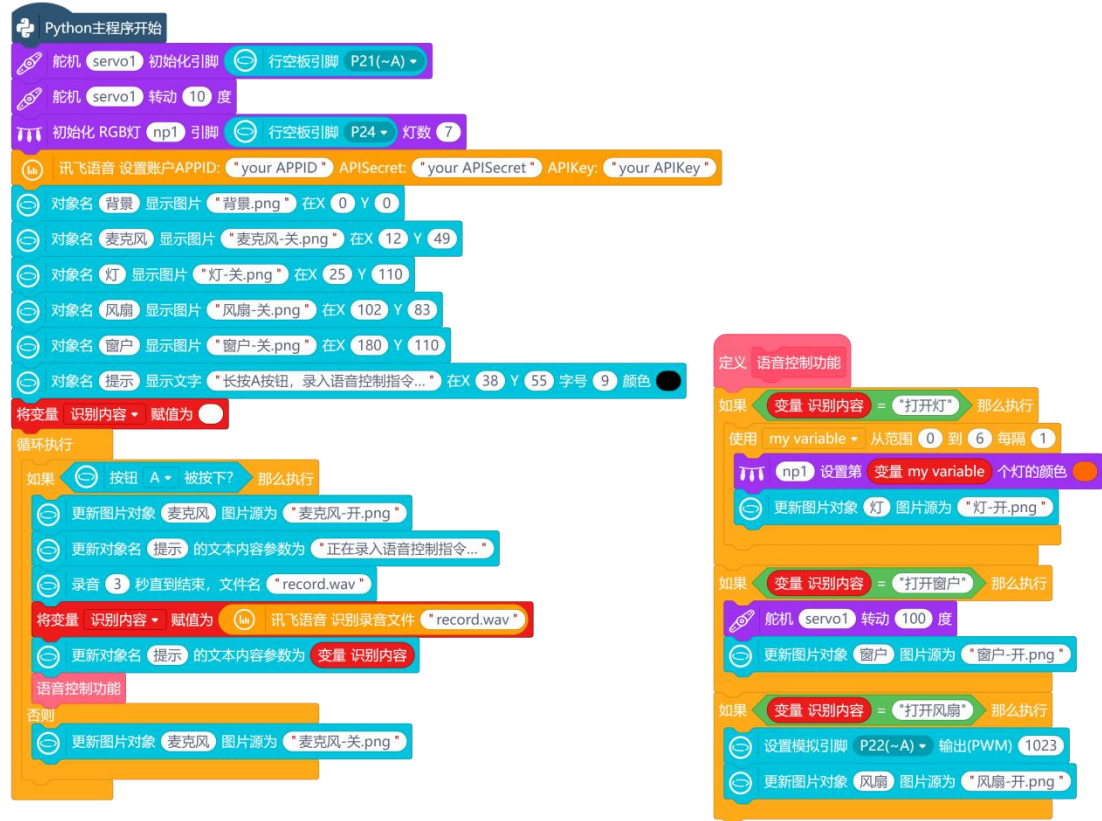
- (5) 当变量识别内容为“打开窗户”时，使用舵机转动指令，控制舵机转动到 100 度。然后使用更新图片对象指令，更新窗户对象的图片源为“窗户-开.png”。



- (6) 当变量识别内容为“打开风扇”时，使用设置引脚输出 PWM 指令，设置风扇模块输出的 PWM 值为 1023。然后使用更新图片对象指令，更新风扇对象的图片源为“风扇-开.png”。

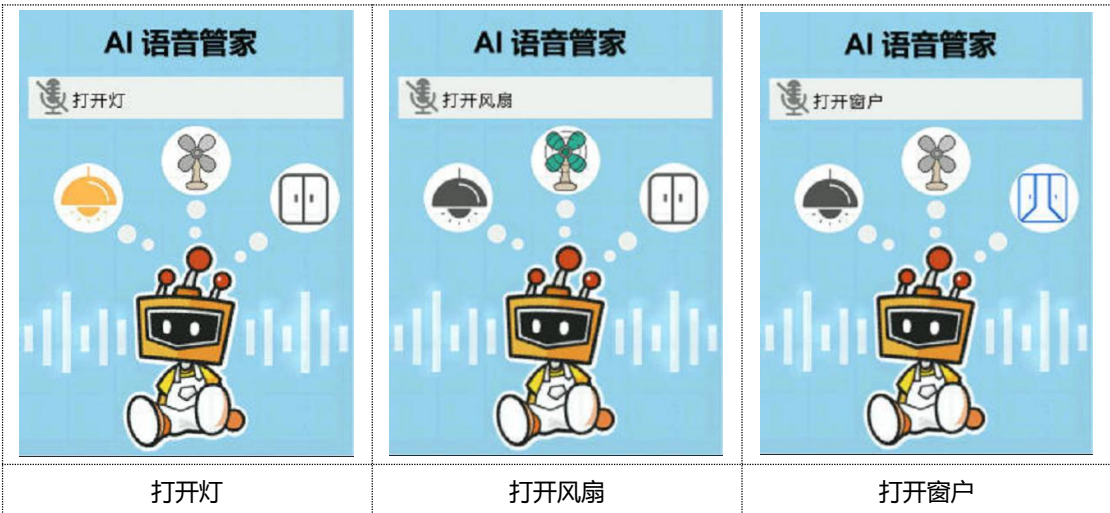


4. 参考程序



5. 程序运行

点击右上角的运行按钮，程序运行成功后，长按 A 按钮，对着行空板的麦克风录入语音控制指令“打开灯”，语音识别成功后，RGB 灯带打开，并将行空板上的灯状态图切换为开。打开风扇、打开窗户的操作，使用同样的方式即可。



知识园地

1. 什么是语音识别技术

语音识别技术，也被称为自动语音识别（Automatic Speech Recognition，简称 ASR）是让机器通过识别把语音信号转变为相应的文本或命令的人工智能技术，也就是让机器听懂人类的语音的技术。它的

基本工作流程是，将录入的声音进行数字化和预处理之后，通过特征提取获得能够表征语言特点的特征向量，然后通过加入模型库进行模式匹配，获得概率最高的文本，输出最终结果。



2. 了解语音指令控制硬件工作的方法

通过语音指令如何控制硬件工作呢？

当输入的语音指令，被识别成功后，将识别结果和设定的语音控制指令进行判断，如果识别结果和语音控制指令判断结果为真时，控制执行传感器执行对应的操作。

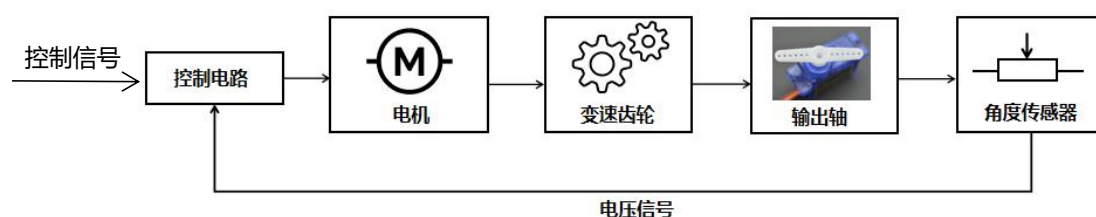


3. 了解舵机工作原理

舵机又称伺服电机，它是一种带有输出轴的装置。它主要由输出轴、变速齿轮、角度传感器、直流电机、控制电路这几部分组成。



舵机是如何工作的呢？舵机的控制电路接收到角度控制信号后，控制电机转动，电机带动变速齿轮组，减速后传至输出轴。



如何才能知道，舵机是否转动到指定的角度呢？舵机的输出轴和角度传感器是相连的，输出轴转动的同时，会带动角度传感器，角度传感器会对控制电路进行电压信号的反馈，然后控制电路再次根据所在位

置决定电机的转动方向和速度，从而达到舵机输出指定角度。

4. 指令学习

Block	功能说明
	该指令用于录制语音，使用时需要写明录制时长以及音频存储的文件名。
	该指令用于讯飞语音服务设置。使用时，需要写明账号、密码和密钥。
	该指令用于获得录音文件的识别文本。使用时需要写明要识别的录音文件。
	该指令用于初始化舵机连接到行空板的引脚。
	该指令用于设置舵机转动角度。

挑战自我

任务三的程序，实现了使用语音控制指令，控制灯带、舵机、风扇的开启工作。接下来，大家在任务三程序的基础上，完成使用语音控制指令控制设备关闭的程序！



第 8 课 AI 电子警察

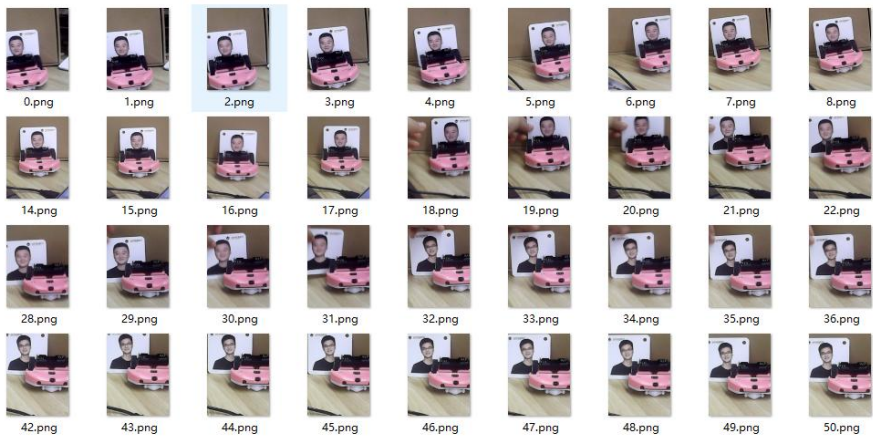
近年来，人工智能技术的发展日新月异，为解决这一难题提供了新的解决方案，那就是 AI 电子警察。AI 电子警察通过智能化的交通管理手段，能够高效、准确地识别行人与车辆违规行为，提高交通管理的效率和质量，为城市交通安全保驾护航。

那么，你是否对 AI 电子警察感到好奇？是否想了解它的工作原理和应用场景？接下来，就使用行空板、USB 摄像头制作一款 AI 电子警察。帮助人们文明出行。



任务目标

摄像头模拟电子眼，如果摄像头同时检测到有车与人脸时，就对该画面进行抓拍，并将抓拍的照片保存到行空板的系统文件里。



知识点

1. 掌握 USB 摄像头的工作原理及使用方法。
2. 掌握行空板查看系统文件的方法。

- 3. 了解物体检测技术。
- 4. 了解 OpenCV。

材料清单

硬件清单：

	
行空板 x1	USB 摄像头 x1
	
白色硅胶绞线 x1	Type-C USB 线 x1

软件使用：Mind+编程软件 x1
下载地址：<https://mindplus.cc/>

动手实践

AI 电子警察主要用于抓拍车辆不礼让行人的违规行为。如果摄像头同时检测到车和人脸就对该画面进行抓拍，然后将抓拍的照片保存在行空板的系统文件里，后期可以调取里面的照片筛选违规车辆。接下来，通过三个简单的小任务来学习如何制作 AI 电子警察的项目。

任务一：显示实时视频

在此项任务中，打开摄像头，并将摄像头检测到的画面显示在行空板屏幕上。

任务二：完成人脸与车辆检测功能

在任务一的基础上，加上人脸与车辆检测的功能，当检测到人脸时，就在终端区域输出“检测到行人”的信息；当检测到汽车时，就在终端区域输出“检测到有车辆”的信息。

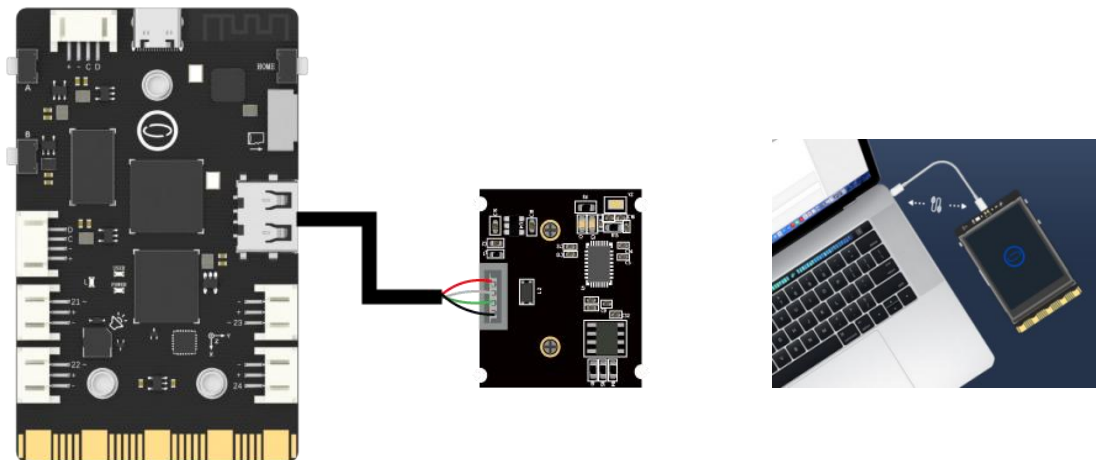
任务三：制作 AI 电子警察

如果摄像头同时检测到有车与人脸时，就对该画面进行抓拍，并将抓拍的照片保存到行空板的系统文件里。

任务一：显示实时视频

1. 硬件连接

将 USB 摄像头连接到行空板上，如下图。硬件连接成功后，使用 USB 线将行空板连接到计算机。

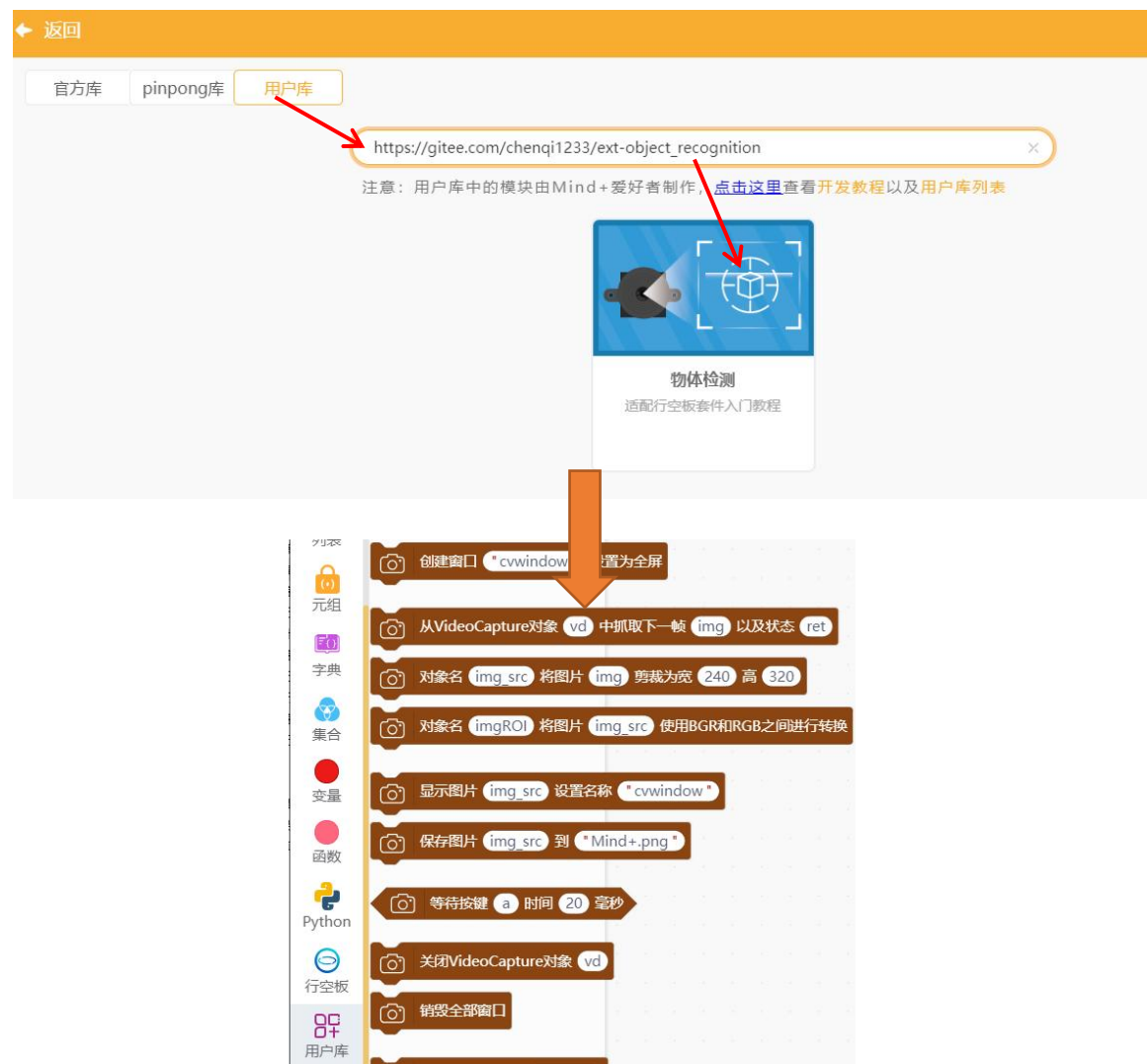


2. 软件准备

(1) 打开 Mind+, 按照下面图示完成软件准备过程。



(2) 点击“扩展”，在用户库的搜索框输入地址“https://gitee.com/chenqi1233/ext-object_recognition”然后按下回车按键搜索“物体检测”的库，搜索完成后点击“物体检测”库完成添加。

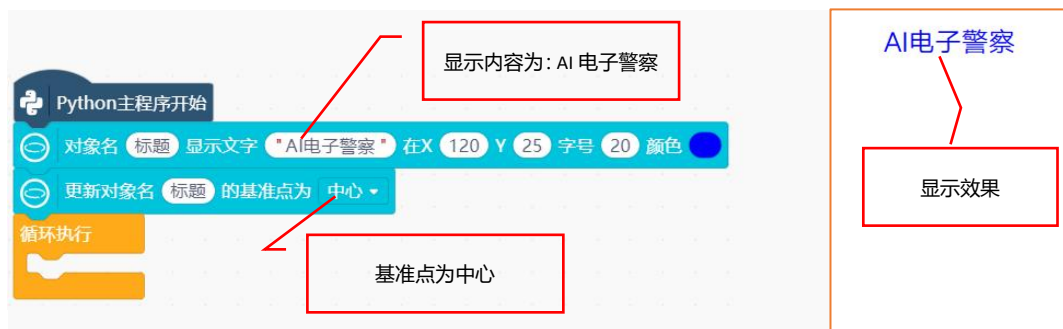


3. 编写程序

首先观察一下，任务一的界面上都包含哪些元素？标题，显示框，视频内容显示区域，效果如下图。



- (1) UI 界面设置，首先利用“显示文字”指令将对象“标题”的内容显示在行空板屏幕上方。



- (2) 使用“显示圆角矩形”指令在坐标 (0,50) 的位置画一个宽为 238、高为 268、圆角半径为 5、线宽为 3 的圆角矩形边框。



- (3) 想要显示实时的视频画面，就需要使用“初始化摄像头”指令打开摄像头，并设置该摄像头的设备号为 0。



- (4) 现在摄像头已经打开了，接下来我们可以使用“从 VideoCapture 对象 (vd) 中抓取下一帧 (img) 以及状态 (ret)”指令将捕获到的画面存入到 ‘img’ 这个变量中，然后使用“对象名 (img_src) 将图片 (img) 裁剪为宽 () 高 ()”指令将 ‘img’ 变量里的图像裁剪成宽为 240，高为 270 的新图像，并新图像保存在 ‘img_src’ 变量中，以供后续处理和分析使用。

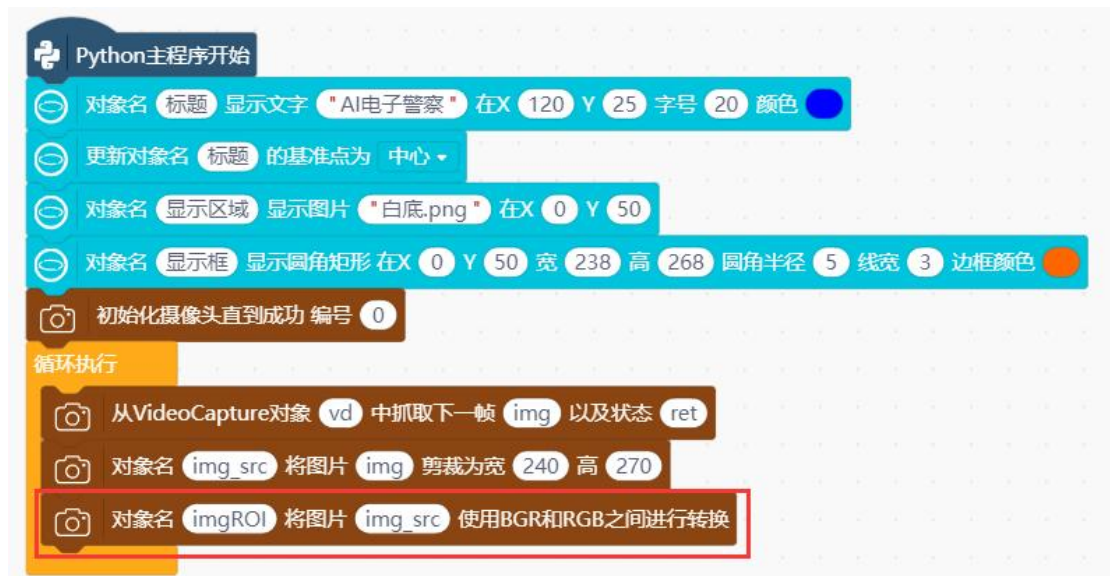


- (5) 上面我们已经学习了如何捕获画面和裁剪画面，接下来，我们将完成画面实时显示功能。首先导入一张白底的背景图到文件系统中，然后在 (0,50) 的坐标位置显示这张背景图。



- (6) 因为 OpenCV 图像中的默认颜色格式是 BGR，想要在行空板屏幕上显示画面，就需要将颜色格式转换成 RGB，使用“对象名 (imgROI) 将图片 (img_src) 使用 BGR 和 RGB 之间进行转换”指令将 'img_src' 变量中的图像转换成 RGB，并将转换后的图像保存到 'imgROI' 变量中。

注意：关于“openCV”的相关介绍，见知识园地。



- (7) 通过上面六项步骤, 我们已经完成了图像颜色格式转换、图像尺寸剪裁和图像捕获。接下来就利用“更新图片”的方法将捕获到的图像显示在显示框区域内。



4. 参考程序



5. 程序运行

检查行空板是否已连接到 mind+, 无误后, 点击右上角的运行按钮, 程序运行成功后, 行空板屏幕上会显示实时画面。



任务二：完成人脸与车辆检测功能

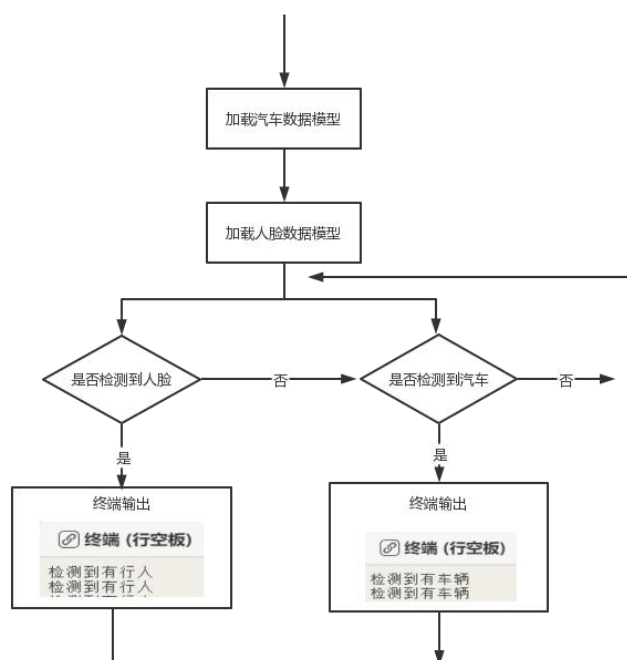
前面我们讲过, 在任务一的基础上, 加上人脸与车辆检测的功能, 当检测到人脸就在终端区域输出“检测到有行人”的信息; 当检测到汽车就在终端区域输出“检测到有车辆”的信息。

1. 准备工作

按照任务一的“[硬件连接](#)”与“[软件准备](#)”连接好行空板并准备好 mind+软件。

2. 编写程序

在程序开始之前, 先分析一下需要任务二需要完成哪些功能? 首先要加载汽车与人脸数据模型, 然后再判断检测到的对象是不是汽车或人脸, 如果摄像头检测到是汽车时, 终端区域输出“检测到有车辆”; 如果摄像头检测到是人脸, 终端区域输出“检测到有行人”。



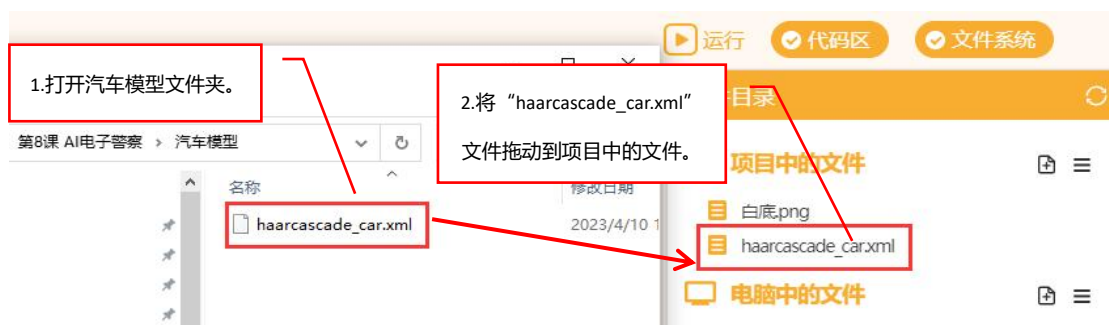
- 由上图得知，想要实现人脸检测的功能，需要提前加载人脸数据模型。首先利用“创建级联分类器 (casccade)”指令创建一个空的级联分类器对象，并将其赋值给变量 ‘casccade’。由于创建的级联分类器对象是空的，无法进行物体检测。需要通过“对象名 (casccade) 加载 [人脸 (默认) 数据模型] 模型”指令加载一个训练好的人脸检测模型文件。这样，‘casccade’ 就可以用来检测画面中的人脸了。



- 人脸数据模型加载完之后，需要利用“对象名 (casccade) 是否加载完成”指令来检查分类器是否已经加载了一个训练好的分类器模型。



- 完成上面两项步骤后，接下来就是加载汽车数据模型。和加载默认的人脸数据模型不同的地方是我们需要提前将汽车数据模型导入到文件系统上的“项目中的文件”，步骤如下：



注意：可自己去网上下载汽车数据模型（后缀为 .xml 的文件），下载成功后，需要将文件名修改成“haarcascade_car.xml”。

汽车数据模型导入成功后，根据（1），（2）项的步骤完成汽车数据模型加载。



- (4) 当确定分类器 'casceadecar' 与 'casceadecar' 已经加载好了人脸与汽车数据模型后，就利用“对象名 (casceadecar) 输入检测图像 (img) 返回检测结果信息”指令对 'img' 变量里的图像进行物体检测，然后返回检测结果。为了方便后续处理和分析，我们需要创建两个变量 "face"，"car" 用来存储检测结果。



- (5) 运行步骤（4）的代码，观察终端区域的运行结果你会发现，当摄像头检测到汽车时，会输出一串数字类型的数据，未检测到汽车时，输出的是一个 "()"。因为检测到汽车，输出的是一串数字，所以我们可以判断变量 car, face 数据的长度，来判断画面中是否有汽车或人脸。当变量 car, face

数据的长度 > 0 时，说明画面中检测到人脸或汽车了，这时终端区域就输出“检测到有行人”或“检测到有车辆”。



3. 参考程序



4. 程序运行

点击右上角的运行按钮，程序运行成功后，你会看见，当摄像头检测到是人脸，终端区域输出“检测到有行人”，当摄像头检测到是汽车时，终端区域输出“检测到有车辆”。

检测到人脸		终端 (行空板) 检测到有行人 检测到有行人 检测到有行人 检测到有行人 检测到有行人 检测到有行人 检测到有行人 检测到有行人
检测到汽车		终端 (行空板) 检测到有车辆 检测到有车辆 检测到有车辆 检测到有车辆 检测到有车辆 检测到有车辆 检测到有车辆 检测到有车辆

5. 试一试

尝试着去网上下载其他的汽车数据模型，并将它导入到项目文件中，然后用自己寻找的汽车数据模型来完成任务二的项目。

任务三：制作 AI 电子警察

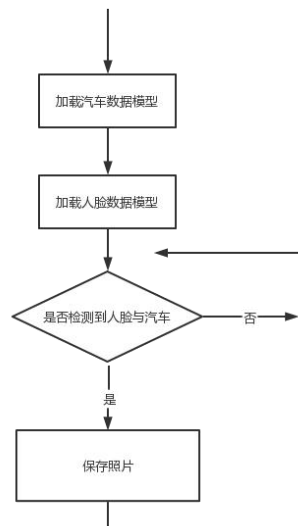
通过任务二，我们已经成功地完成了人脸与车辆检测的功能。接下来，我们将制作一个完整的 AI 电子警察项目。

1. 硬件连接

按照任务一的“[硬件连接](#)”与“[软件准备](#)”连接好行空板并准备好 mind+软件。

2. 编写程序

在编写程序前，先梳理一下任务三需要完成什么功能？在任务二的基础上，完成多物体检测的功能，即同时检测画面中是否有人与汽车。当摄像头检测到画面中有人与汽车时，就对该画面进行抓拍，并将抓拍的照片保存到行空板的系统文件里。



- (1) 由流程图得知，当摄像头同时识别到人脸与汽车时，就将抓拍到的照片保存到行空板的系统文件里。首先我们利用判断条件同时判断变量 `car` 与 `face` 数据的长度都大于 0，如果满足该条件就使用 “保存图片 (`img_src`) 到 (“Mind+.png”)” 指令将抓拍到的图片保存到行空板的系统文件里。



- (2) 因为步骤 (1) 运行后每次检测到新的汽车和人都会覆盖上一张照片，如果想要保存更多的抓拍照片就需要修改程序，实现每抓拍一次就保存一个新的图片名称。首先新建一个变量 “my variable” 用来更新照片的名称。



然后在判断条件中将变量 `my variable` 加 1，所得到的结果为 `my variable=my variable+1`。在后面保存图片时都会保存一个新名称，就不会覆盖上一张照片了。



- (3) 步骤 (2) 已经实现了如何更新照片名称，接下来还是使用 “保存图片 (img_src) 到 (“Mind+.png”)” 指令将每张照片保存到行空板的系统文件里。

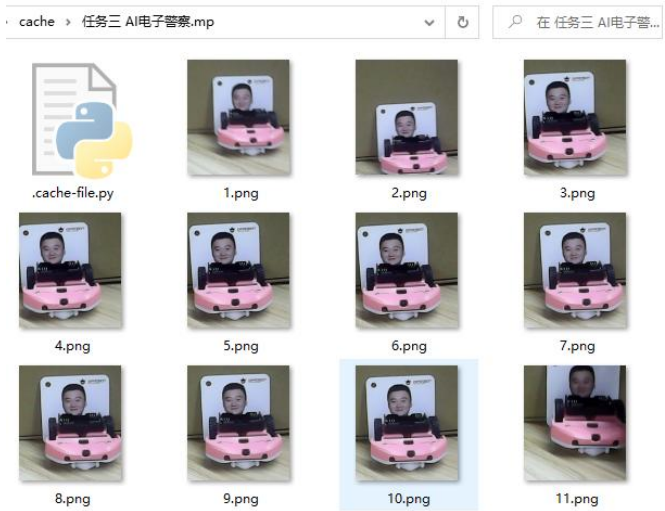


3. 参考程序



4. 程序运行

点击右上角的运行按钮，程序运行成功后。摄像头同时检测到人脸与汽车时，就抓拍该画面，并将抓拍到的照片保存到行空板的系统文件里。如何查看抓拍到的照片，可以参考知识园地的方法。



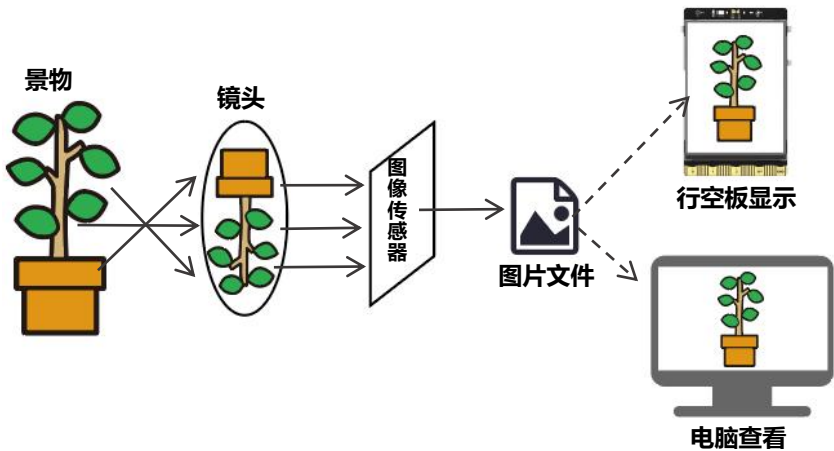
知识园地

1. 认识 USB 摄像头

这节课中，用到的摄像头是一款免驱动的 USB 摄像头，只要将摄像头插入 USB 端口，就可以工作起来。



摄像头是如何工作的呢？下面简单地了解一下摄像头的工作原理。景物通过光学镜头生成图像并投射到摄像头内部的图像传感器上，并对图像信号进行加工处理后变成图片文件，再通过 USB 接口将图片传输到行空板中，最后我们可以在行空板屏幕上显示图片，也可以通过将图片文件下载到电脑上，查看图片。

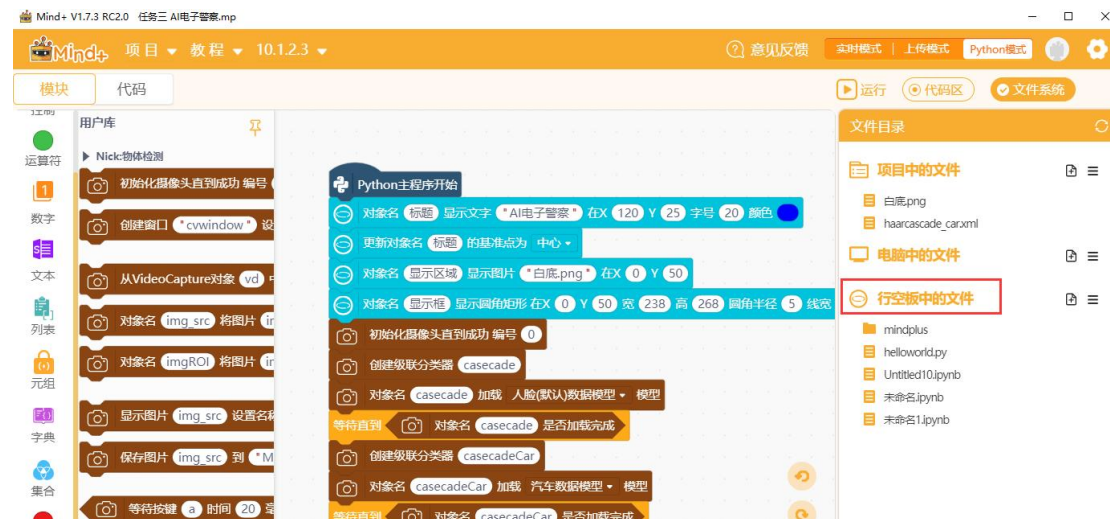


2. 行空板查看系统文件的方法

行空板是一种嵌入式 Linux 系统开发板，它内置了 Samba 服务。可以让用户方便地在本地网络中访问行空板上的共享文件夹，进行文件的传输和共享。

1) 如何查看保存到行空板的图片

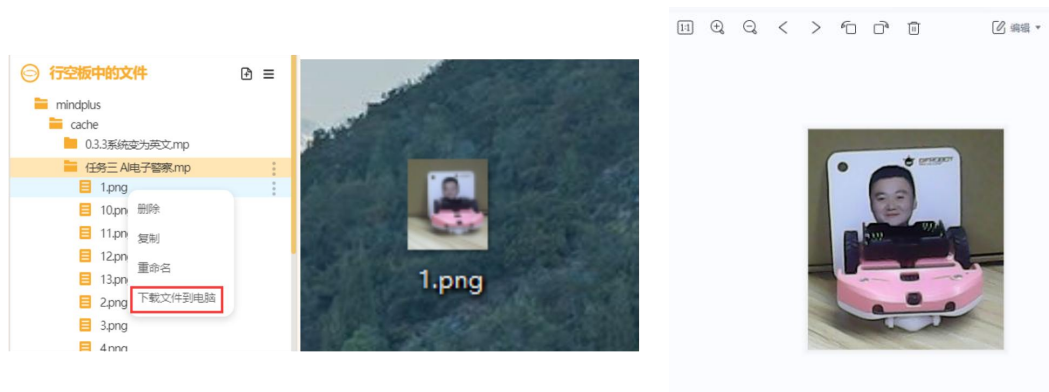
在任务三中，用到了保存图片指令，使用这个指令保存图片具体存到了什么位置呢？连接行空板 10.1.2.3 远程终端后，可以看到在文件系统的文件目录下，会多出一个名为“行空板中的文件”。



点击目录中的 mindplus 文件夹，然后再点击 cache 文件夹，找到“任务三 AI 电子警察.mp”文件的保存位置。点击“任务三 AI 电子警察.mp”，可以看到抓拍的图片文件分别为：1.png、2.png、3.png.....。



其中 1.png、2.png、3.png.....就是使用保存图片指令从摄像头中获取的图片。右击“xx.png”，选择“下载文件到电脑”，将图片下载到电脑上。记住保存的路径，通过电脑就可以查看摄像头拍摄的图片了。



2) 如何用计算机访问行空板系统文件

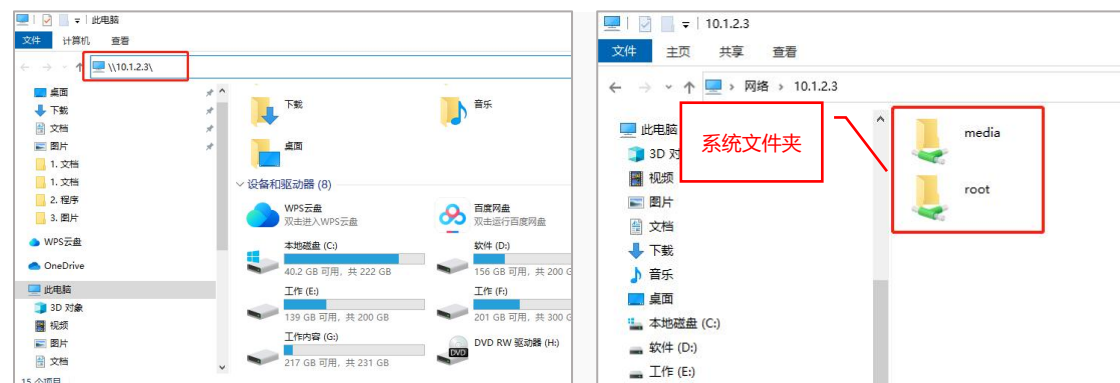
开启行空板文件共享

长按行空板的“Home”键进入菜单页，点击“3-应用开关”，检查文件共享服务是否开启，如果显示“已禁用”，点击文件共享选项，切换文件共享状态为“已启用”。

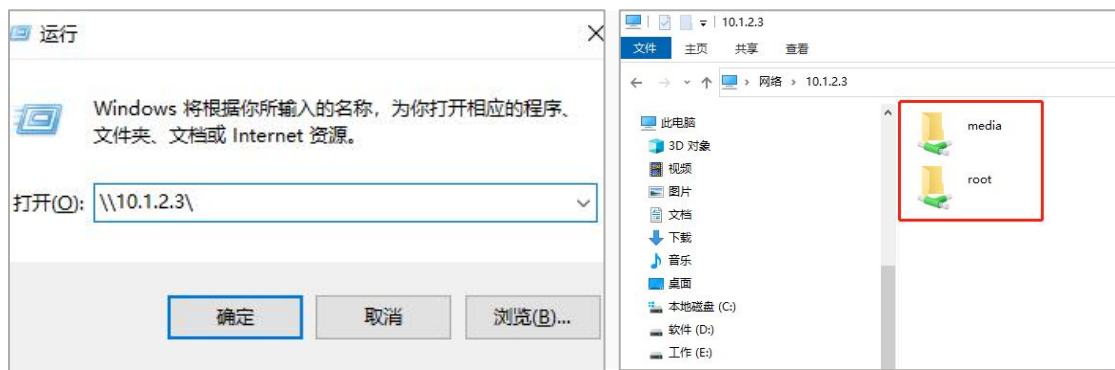


配置电脑

方法 1: 使用 USB 线将行空板连接到计算机，然后在电脑桌面上找到“此电脑”双击打开。在地址栏输入行空板的网络地址，格式为\\ip 地址\。如下图，输入\\10.1.2.3\之后按回车，打开行空板系统文件夹。



方法 2: 使用快捷键 win+R 打开运行窗口，输入网络地址\\10.1.2.3\，点击确定，打开行空板系统文件夹。



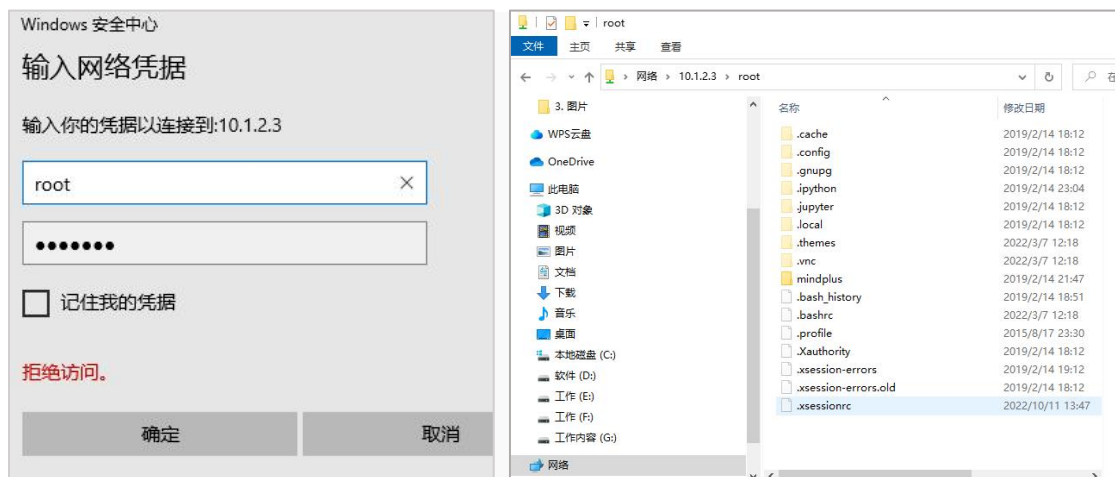
行空板系统文件夹中，分别有 media、root 文件夹，其中，root 为行空板板载的内存空间，media 为外部连接的硬盘空间（给行空板插上 SD 卡或者通过 USB 口连接硬盘后会自动挂载在这个文件夹下）。

查看保存到行空板的图片

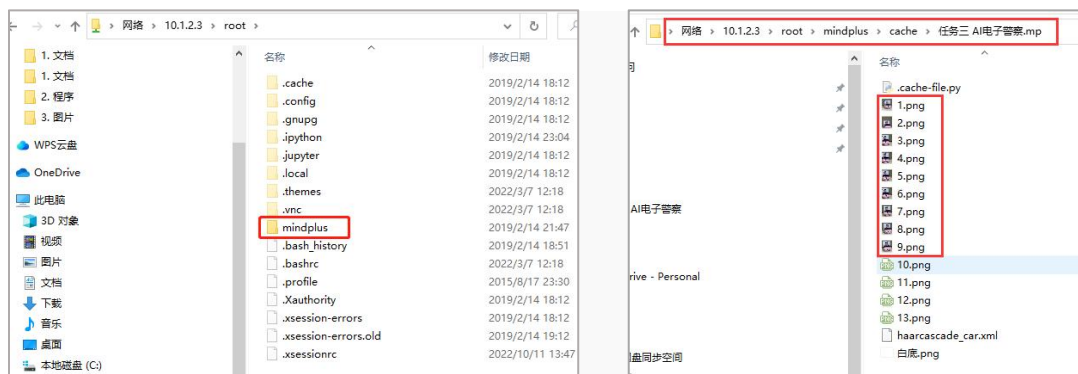
双击 root 文件夹，会提示输入访问账号和密码。输入对应的账号和密码后，点击确定进入行空板的 root 文件夹。

账号：root

密码：dfrobot



在文件夹列表中，找到 mindplus 文件夹，然后再点击 cache 文件夹，找到“任务三 AI 电子警察.mp”文件的保存位置。点击“任务三 AI 电子警察.mp”，同样可以看到抓拍的图片文件分别为：1.png、2.png、3.png.....。



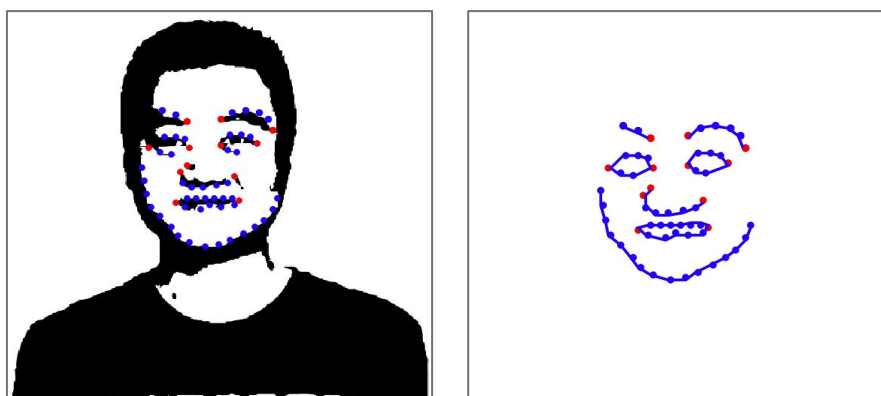
3. 了解 OpenCV

当我们处理图像和视频时，通常需要进行各种操作，如图像分割、目标检测、人脸识别、图像滤波等。OpenCV (Open Source Computer Vision Library) 是一种计算机视觉技术，它可以帮助我们对图像和视频进行各种操作。例如：图像处理；特征检测；目标检测和跟踪；三维重建；机器学习；视频分析；图像和视频的显示等。

在本项目中，我们实现了人脸和车辆检测的功能，这两种功能都是基于 OpenCV 中的物体检测技术。在 OpenCV 中实现物体检测的技术主要有两种：基于特征的方法和深度学习方法。项目中主要使用的是基于特征的方法。基于特征的物体检测方法是将图像中的目标物体和背景分开，并将目标物体的特征提取出来。这些特征通常是目标物体的**形状、颜色、纹理**等。例如要检测人脸，则需要提取人脸的边缘、轮廓、纹理等，如下图。



一旦目标物体的特征被提取出来，就可以使用机器学习算法对其进行分类，以检测目标物体。基于特征的物体检测方法通常使用 Haar Cascade 和 HOG (Histogram of Oriented Gradients) 算法。Haar Cascade 是一种基于特征的目标检测算法，它可以对人脸、车辆、眼睛等进行检测。HOG 算法则是一种用于人体检测的算法，它可以对行人、手势等进行检测。项目中主要使用的是 Haar Cascade 算法。在使用 Haar Cascade 算法进行对象检测时，首先需要使用大量的正样本和负样本训练得到一个 Haar 分类器模型。然后加载该分类器模型并对人脸提取出的特征进行判别，分析出图像中是否存在人脸。



项目中讲解了物体检测技术的基本操作。如果您想进一步了解 OpenCV 库的功能和应用，请查看[OpenCV 官方文档](#)。

4. 指令学习

动手实践里主要使用了以下操作指令。

Block	功能说明
 对象名 显示圆角矩形 在X 0 Y 0 宽 100 高 200 圆角半径 5 线宽 1 边框颜色	该指令用于在行空板上显示空心圆角矩形,在指令中可以设置圆角矩形的位置、宽、高、圆角半径、线宽以及边框颜色。
 初始化摄像头直到成功 编号 0	该指令的作用是打开摄像头,并等待摄像头设备初始化成功,编号表示摄像头设备号,参数为 0 表示使用摄像头设备进行视频捕获。
 创建级联分类器 cascade	该指令用于创建一个级联分类器对象 (cascade), 主要用于加载已训练好的分类器模型文件,并进行物体检测。
 对象名 cascade 加载 人脸(默认)数据模型 模型 人脸(默认)数据模型 汽车数据模型	该指令用于加载已经训练好的级联分类器模型文件,例如默认的人脸数据模型与汽车数据模型。
 对象名 cascade 是否加载完成	该指令的作用是检查级联分类器对象 (cascade) 是否已经成功加载了分类器模型文件。如果返回的结果为 True, 表示可以继续后续进行后续的物体检测操作, 否则无法进行后续的物体检测操作。
 从VideoCapture对象 vd 中抓取下一帧 img 以及状态 ret	该指令的作用是从 VideoCapture 对象 vd 所表示的摄像头中读取一帧图像,并将读取到的图像存储在 img 变量中。同时,返回值 ret 表示读取图像的操作是否成功,如果 ret 为 True, 则说明成功读取到了一帧图像;如果 ret 为 False, 则说明读取图像失败。
 对象名 img_src 将图片 img 剪裁为宽 240 高 270	该指令的作用是将 'img' 变量里的图像剪裁成宽为 240, 高为 270 的新图像,并新图像保存在 'img_src' 变量中,以供后续处理和分析使用。
 对象名 imgROI 将图片 img_src 使用BGR和RGB之间进行转换	该指令的作用是将图像 img_src 从 BGR 格式转换为 RGB 格式,并创建了一个 PIL 库中的 Image 对象 imgROI, 以便后续对图像进行处理和显示。
 对象名 cascade 输入检测图像 img_src 返回检测结果信息	该指令的作用是检测输入图像中的对象,并返回对象的矩形区域坐标结果信息。
 保存图片 img_src 到 "Mind+.png"	该指令的作用是将图像 img_src 保存为一个文件, 文件名为 "Mind+.png", 保存的格式为 PNG。

挑战自我

仔细观察生活中的交通,你会发现每个路口都要一个红绿灯。接下来,大家修改一下程序,在行空板

屏幕上设计一个交通信号灯，红灯和绿灯显示的时间为 15 秒，黄灯显示的时间为 5 秒。

提示：需要使用线程和函数。



第 9 课 AI 趣味跟随相机

AI 趣味跟随相机是一种能够自动跟踪拍摄对象的相机，它使用人工智能技术和计算机视觉技术来实现自动识别和跟踪。它可以根据人脸的坐标，自动调整角度，确保拍摄过程中都有人脸出现。让我们拍摄过程变得更加简单、轻松、有趣，每个人都可以是摄影达人。

接下来，就使用行空板、USB 摄像头、舵机，制作一款可以根据人脸坐标自动调整角度的 AI 趣味跟随相机吧！



任务目标

摄像头检测到人脸后，获取人脸的坐标值。根据人脸坐标值，控制舵机的旋转角度，实现摄像头跟随人脸移动。

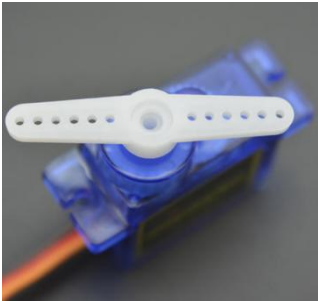


知识点

1. 了解人脸检测技术的实现原理。
2. 学习使用 opencv 库进行人脸检测的方法。
3. 学习如何根据坐标判断人脸的位置。
4. 掌握根据人脸坐标调节舵机角度的方法。

材料清单

硬件清单:

	
行空板 x1	USB 摄像头 x1
	
舵机 x1	Type-C USB 线 x1

软件使用: Mind+编程软件 x1

下载地址: <https://mindplus.cc/>

动手实践

AI 趣味跟随相机, 主要是让 USB 摄像头跟随人脸自动调整角度, 实现摄像头跟随人脸, 当人脸处于图像中间位置时, 就拍照并存储。接下来, 就从下面两个小任务来完成项目吧!

任务一: 检测并框选人脸

将摄像头拍摄到的视频图像, 实时的显示在行空板屏幕上。根据摄像头中的图像, 判断是否出现人脸。如果出现人脸, 使用矩形框将人脸框选出来。

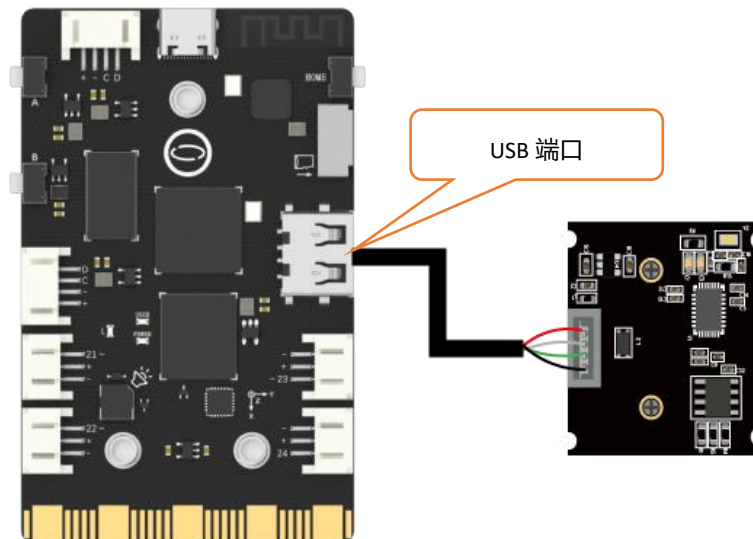
任务二: 制作人脸跟随相机

通过获取的人脸坐标值, 控制舵机转动角度, 让摄像头跟随人脸一起移动。当人脸在出现在图像中间位置时, 就拍下照片并将照片保存到行空板本地。

任务一: 检测并框选人脸

1. 硬件连接

使用摄像头专用 USB 线, 将摄像头连接到行空板的 USB 端口。然后, 使用 USB 线将行空板连接到计算机。



2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。



点击“扩展”，在用户库的搜索框输入地址“https://gitee.com/chenqi1233/ext-object_recognition”
然后按下回车按键，找到“物体检测”的库并点击，完成添加。



3. 编写程序

检测人脸

- (1) 要实现显示实时的视频图像，首先需要用到“初始化摄像头直到成功”指令，打开摄像头。摄像头打开后，使用“创建窗口 cvwindow 设置为全屏”指令，设置视频在行空板上全屏显示。



- (2) 然后在“循环执行”指令中，使用“从 VideoCapture 对象 (vd) 中抓取下一帧 (img) 以及状态 (ret)”指令，将捕获的图像存入'img'中。



- (3) 如何才能判断，捕获的图像中，是否有人脸呢？使用“创建级联分类器 (cascade)”指令，创建一个空的级联分类器对象。使用“对象名 (cascade) 加载 [人脸 (默认) 数据模型] 模型”指令，加载一个训练好的人脸检测模型文件。然后，使用“对象名 (cascade) 是否加载完成”的指令，来检查分类器是否已经加载了一个训练好的分类器模型。



框选人脸

- (1) 利用“对象名 (cascade) 输入检测图像 (img) 返回检测结果信息”的指令，对 ‘img’ 中的图像进行人脸检测，然后返回检测结果（返回的结果为检测到的人脸位置信息元组(x, y, w, h)）。为了方便后续处理和分析，新建“变量 face”，用来获取人脸位置信息。



- (2) 如何才能将 face 元组中的人脸坐标信息提取出来呢？首先，新建 4 个变量，变量 x、变量 y、变量 w、变量 h，然后“在 Python 主程序开始”之下，初始化 4 个变量的值为 0。



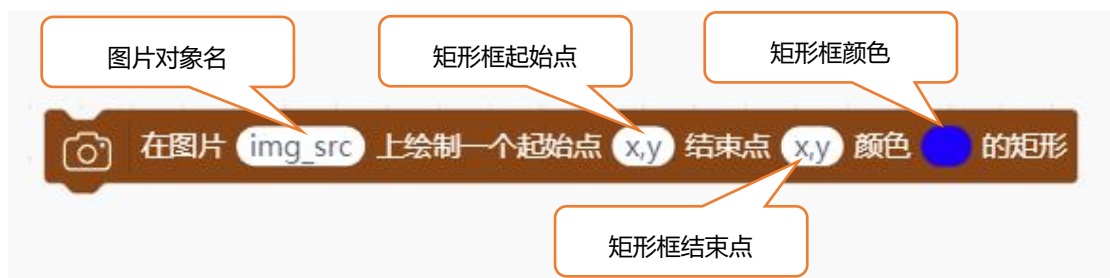
- (3) 使用 “for 循环遍历” 指令，将 face 元组中的[x, y, w, h]值逐个提取出来，分别赋值给对应的变量。

注意：关于 “for 循环” 相关介绍，参考行空板图形化 python 入门教程第 12 课

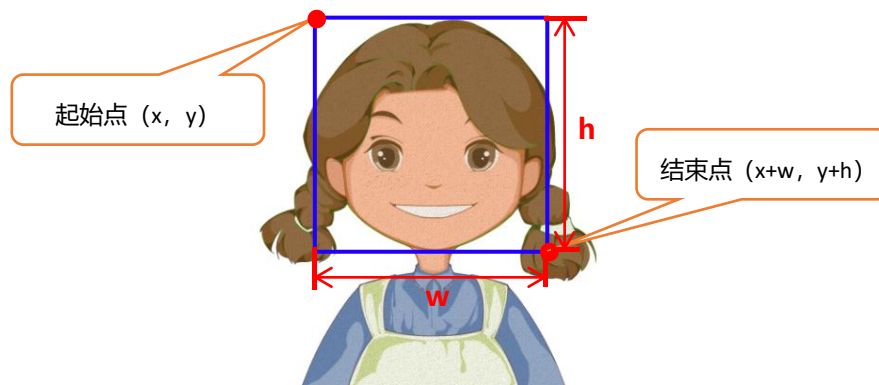
(<https://mc.dfrobot.com.cn/thread-314804-1-1.html>)



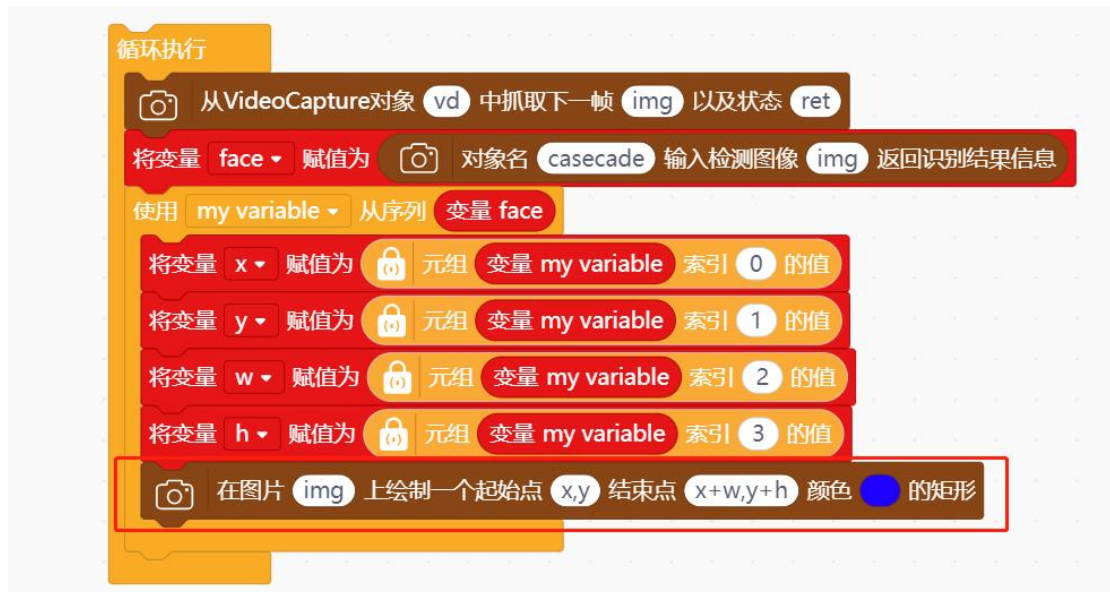
- (4) 获取了人脸位置信息后，使用 “在图片 (img_src) 上绘制一个起始点 (x,y) 结束点 (x,y) 颜色的矩形” 指令，将人脸通过矩形框给框选起来。



- (5) 如何才能确定矩形框的起始点和结束点呢？框选人脸的矩形框，是通过人脸的起始点与结束点来确定的。face 元组中，包含了人脸位置信息的元组[x, y, w, h]，其中，xy 对应的就是检测到人脸的起始点，w 为人脸的宽，h 为人脸的高。因此，通过元组中的数据，就可以确定框选人脸矩形框的起始点坐标与结束点坐标。



- (6) 修改 “在图片 (img_src) 上绘制一个起始点 (x,y) 结束点 (x,y) 颜色的矩形” 指令中的起始点坐标为 (x, y)，修改结束点坐标为 (x+w, y+h)。



- (7) 人脸框选成功后，使用“显示图片 (img) 设置名字 (cvwindow)”指令，将框选后的人脸，显示在行空板上窗口中。

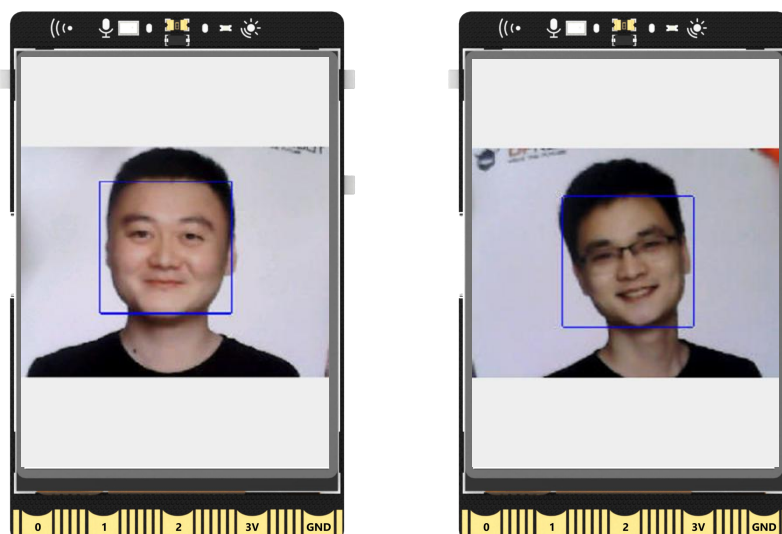


4. 参考程序



5. 程序运行

点击右上角的运行按钮，程序运行成功后，当摄像头中检测到有人脸时，蓝色矩形框自动将人脸框选出来。

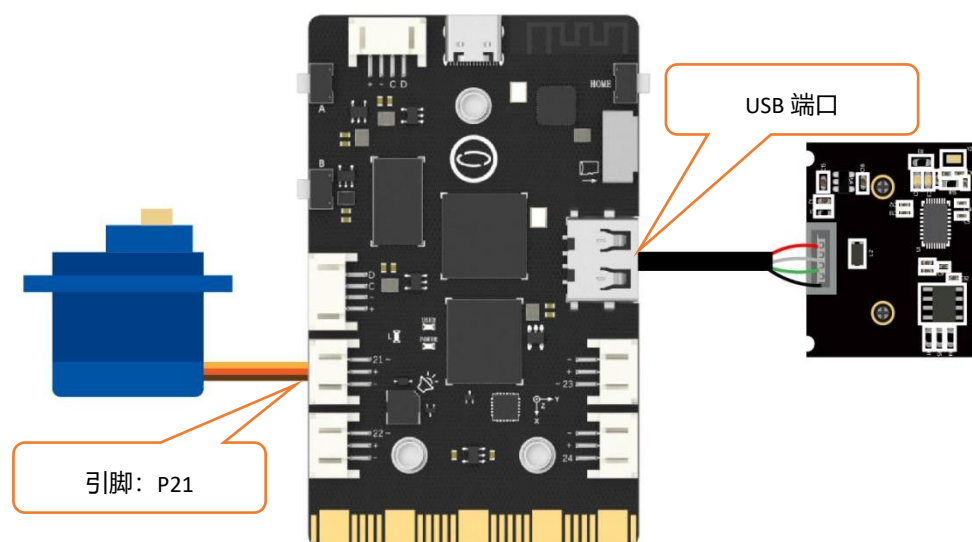


任务二：制作人脸跟随相机

这个任务主要是制作人脸跟随相机，例如，当人脸向左移动时，舵机带着摄像头向左转动；当人脸向右移动时，舵机带着摄像头向右转动；当人脸出现在图像的中间位置时，拍摄照片并将照片保存到行空板本地。

1. 硬件连接

https://gitee.com/chenqi1233/ext-object_recognition



2. 软件准备

- (1) 这个任务，主要是从任务一程序的基础上进行修改。
- (2) 添加舵机库，点击“扩展”，在 pinpong 库中找到“舵机”并点击后，完后添加。

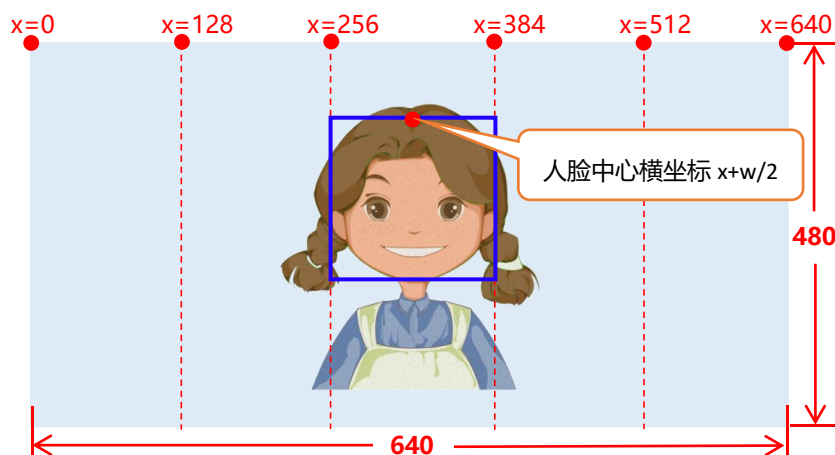


3. 编写程序

- (1) 任务二主要是通过控制舵机的转动角度，实现摄像头跟随人脸一起移动。因此，在任务一程序的基础上，进行程序的添加。在“Python 主程序开始”之下，使用“舵机初始化引脚”指令，初始化舵机引脚为“行空板引脚 P21”，然后使用“舵机转动”指令，设置舵机的转动角度为 90 度（摄像头居中）。



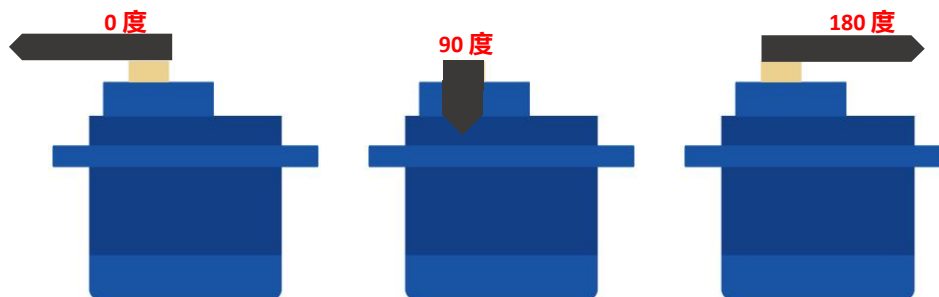
- (2) 如何才能通过判断人脸坐标 x ，让摄像头实现跟随人脸转动呢？首先，我们要知道，摄像头读取到的图像尺寸为 640×480 。需要注意的是，人脸坐标 x 是指人脸在摄像头图像宽（640）中的 x 坐标，而不是行空板上的 x 坐标。



- (3) 当人脸中心横坐标大于 0 且小于 256 时，人脸在图像的左侧，控制舵机带动摄像头向左转动，直到人脸出现在图像的中间位置。新建“变量 **center**”，并“在 Python 主程序开始”下初始化为 0；新建“变量 **angle**”，用来控制舵机角度，并“在 Python 主程序开始”下初始化为 90。



- (4) 首先，需要通过初始化舵机的角度来确定，舵机的转动方向。分别设置舵机的角度为 0 度，90 度，180 度，舵机角度效果如下：



- (5) 变量 center 是用来获取人脸中心横坐标的，利用人脸中心横坐标公式 $(x+w/2)$ ，对“变量 center”进行赋值。使用“条件判断”指令，对变量 center 进行判断 ($center > 0$ and $center < 256$)。当判断条件为真时，使用“舵机转动”指令，控制舵机向右转动 ($angle+1$)。

注意：这里大家可能会有疑惑，当人脸出现在左侧时，为什么舵机要向右转动呢？细心的同学肯定发现了，我们和行空板上显示的图像是呈镜像关系的，当行空板上图像中的人脸向左移动时，其实我们人是在向右移动，因此，需要控制舵机带动摄像头向右转动。



- (6) 当人脸出现在图像右侧时，使用“条件判断”指令，对变量 center 进行判断 ($center > 384$ and $center < 640$)。当判断条件为真时，使用“舵机转动”指令，控制舵机向左转动 ($angle-1$)。



- (7) 当人脸出现在图像的中间位置时，使用“条件判断”指令，对变量 center 进行判断 ($center \geq 256$ and $center \leq 384$)。当判断条件为真时，使用“保存图片 (img) 到 (Mind+.png)”指令，将图片保存到行空板本地。



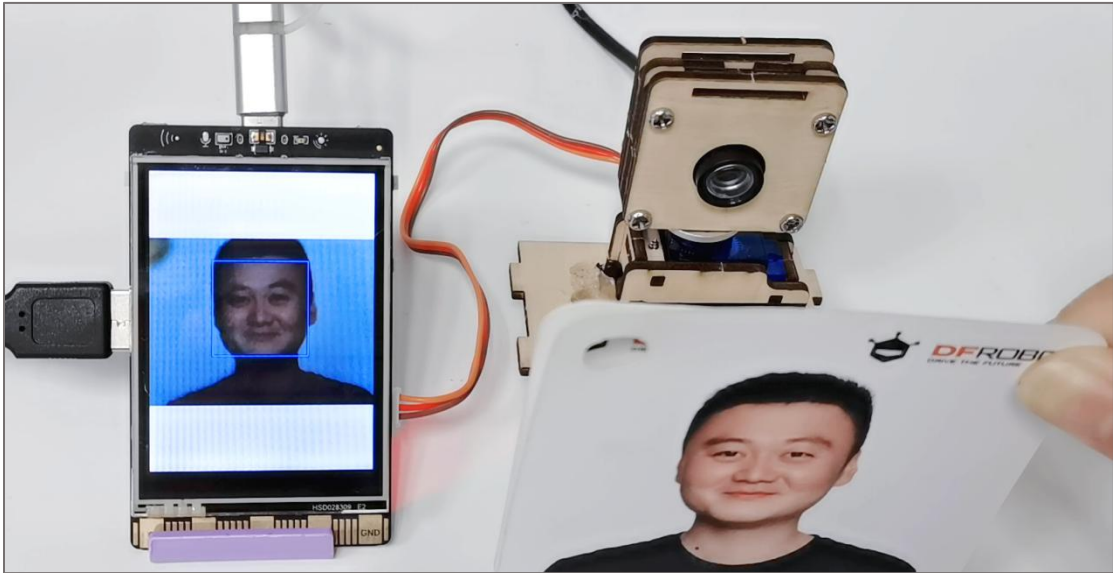
4. 参考程序



5. 程序运行

点击右上角的运行按钮，程序运行成功后。当摄像头中检测到有人脸时，人脸出现在图像中的左侧，舵机带动摄像头向左转动；当人脸出现在图像的右侧时，舵机带动摄像头向右转动；当人脸在图像的中间位置时，拍下照片并将照片保存在行空板本地。

注意：素材文件夹中，提供效果演示的结构文件，可以下载后自行切割。



知识园地

1. 了解人脸检测技术

人脸检测技术是一种计算机视觉技术，用于定位图像或视频中的人脸。该技术通常使用图片处理来检测人脸的位置、大小和方向等。人脸检测技术的实现原理一般包括以下几个步骤：**图像预处理—特征提取—分类器训练—结果输出。**

图像预处理

对输入的图像进行预处理，处理的内容包括图像大小、亮度调整、图像增强、图像二值化等。



原图



处理后

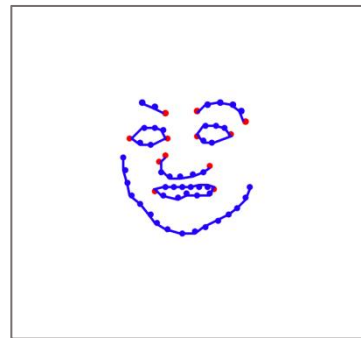
特征提取

通过图像处理和机器学习技术，从图像中提取出可用于识别的特征，例如人脸的边缘、轮廓、纹理等等。



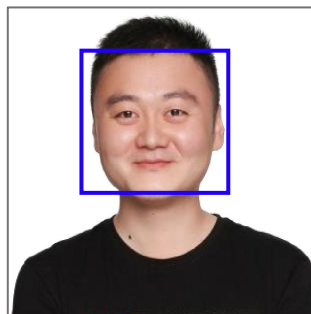
分类器训练

使用 opencv 库中的人脸分类器对人脸提取出的特征进行判别，分辨出图像中是否存在人脸，并确定人脸的位置和大小。



结果输出

将检测结果输出，输出的通常是一个矩形框，用于标记出检测到的人脸的位置和大小。



2. 指令学习

Block	功能说明
 在图片 <code>img_src</code> 上绘制一个起始点 <code>x,y</code> 结束点 <code>x,y</code> 颜色 <code>blue</code> 的矩形	该指令用于在图像上绘制一个矩形框，框出人脸的位置。其中，起始点是矩形框的左上角坐标，结束点是矩形框的右下角坐标。
 使用 <code>my variable</code> 从序列 <code>[0,1,2]</code>	该指令是一个 for 循环指令，用于遍历列表、序列、元组中的每个元素。

挑战自我

任务二中，拍照并将照片保存在行空板本地，照片保存采用的是覆盖的方式，导致行空板的本地文件中，只有一张名为 `Mind+.png` 的图片。下面，大家在任务二程序的基础上进行程序的添加，实现摄像头拍照后采用追加的方式进行对照片进行保存。

提示：保存照片，可以参考第 8 课中保存图片的方法。

第 10 课 IoT 远程控制教室设备

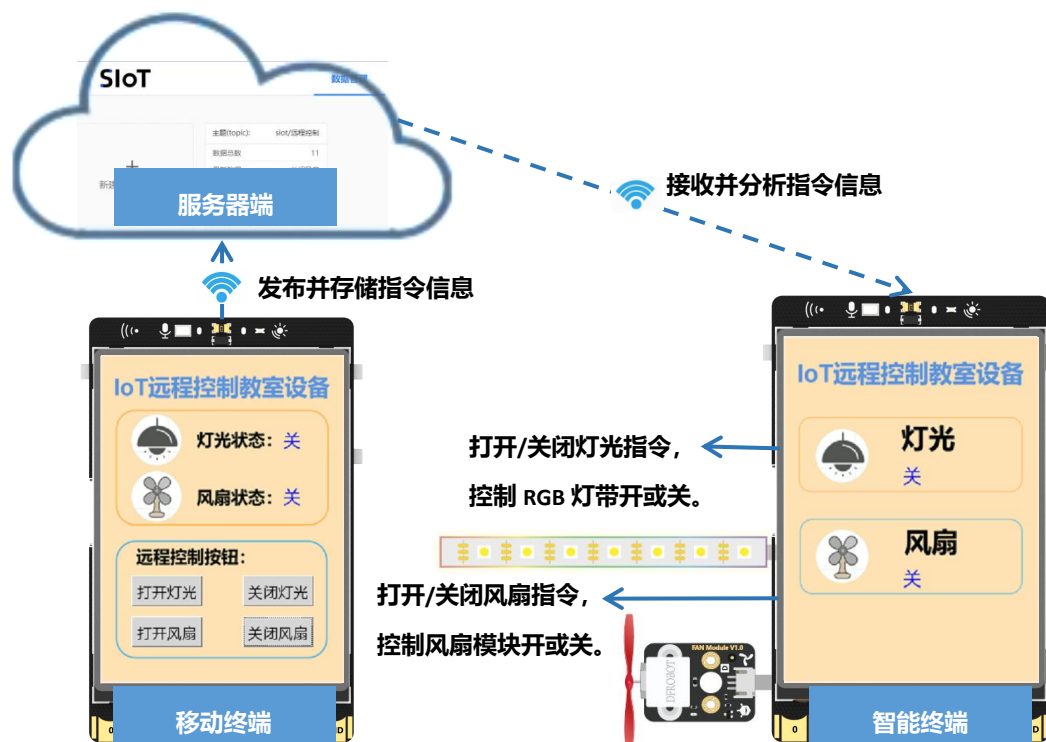
IoT 远程控制教室设备是一种物联网技术的设备控制系统，主要用于学校管理人员，对教室中的设备进行集中监管和控制。接下来，使用行空板、灯带、风扇来制作一款这样的远程控制设备系统，帮助管理人员及时了解教室内设备的使用情况。管理人员不用到教室，只需要通过移动终端上的开关按钮，即可远程控制智能终端上的设备开启和关闭。



任务目标

学校管理人员，可以通过移动终端上的“打开/关闭灯光”按钮和“打开/关闭风扇”按钮，向服务器端发送打开和关闭指令，智能终端接收到指令信息后，根据指令信息控制连接在智能终端上的灯带和风扇设备打开或关闭。

注意：移动终端与服务器端使用的是同一块行空板。

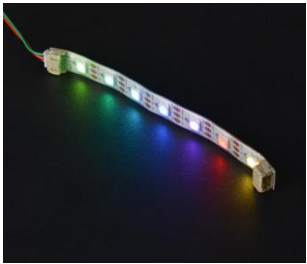
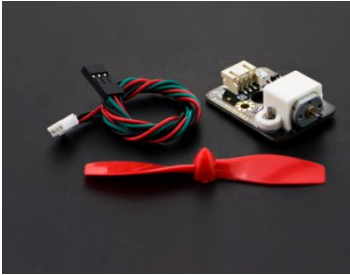


知识点

1. 了解什么是物联网
2. 了解物联网远程控制技术
3. 了解 MQTT 协议
4. 掌握行空板升级 IoT 2.0 服务器的方法

材料清单

硬件清单：

	
行空板 x2	RGB 全彩灯带 x1
	
风扇模块 x1	Type-C USB 线 x2
	
白色硅胶绞线 x2	

软件使用：Mind+编程软件（1.7.3 RC2.0） x1

下载地址：<https://mindplus.cc/>

动手实践

在 IoT 远程控制教室设备这个项目中，学校管理员通过移动终端查看教室设备的工作状态，发现教室

有设备未关闭时，可以直接通过移动终端上的按钮，远程控制智能终端上的设备开启或关闭。这节课，主要从以下三个任务来实现。

任务一：设计移动终端控制界面

这个任务，主要是利用图片素材以及按钮，设计移动终端的控制界面。

任务二：移动终端向 SIoT 发送控制按钮指令

通过移动终端界面上的打开风扇、关闭风扇、打开灯光、关闭灯光按钮，向 SIoT 物联网平台发送对应的开关控制指令。

任务三：制作智能终端

接收 SIoT 物联网平台的设备控制指令，根据对应的控制指令，控制智能终端设备上连接的风扇和灯执行关闭或开启操作。

任务一：设计移动终端控制界面

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

打开 Mind+, 按照下面图示完成软件准备过程。



3. 编写程序

首先观察一下，移动终端控制界面上都包含了哪些元素？背景、灯与风扇图片、设备状态、打开/关闭四个按钮。



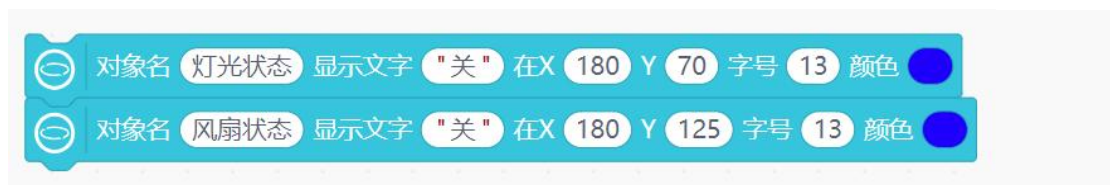
- (1) 设计移动终端控制界面，为了简化程序，将不用改变的文字和背景框都制作成了背景图片。接下来，将图片素材库中的所有图片，都导入文件系统中。



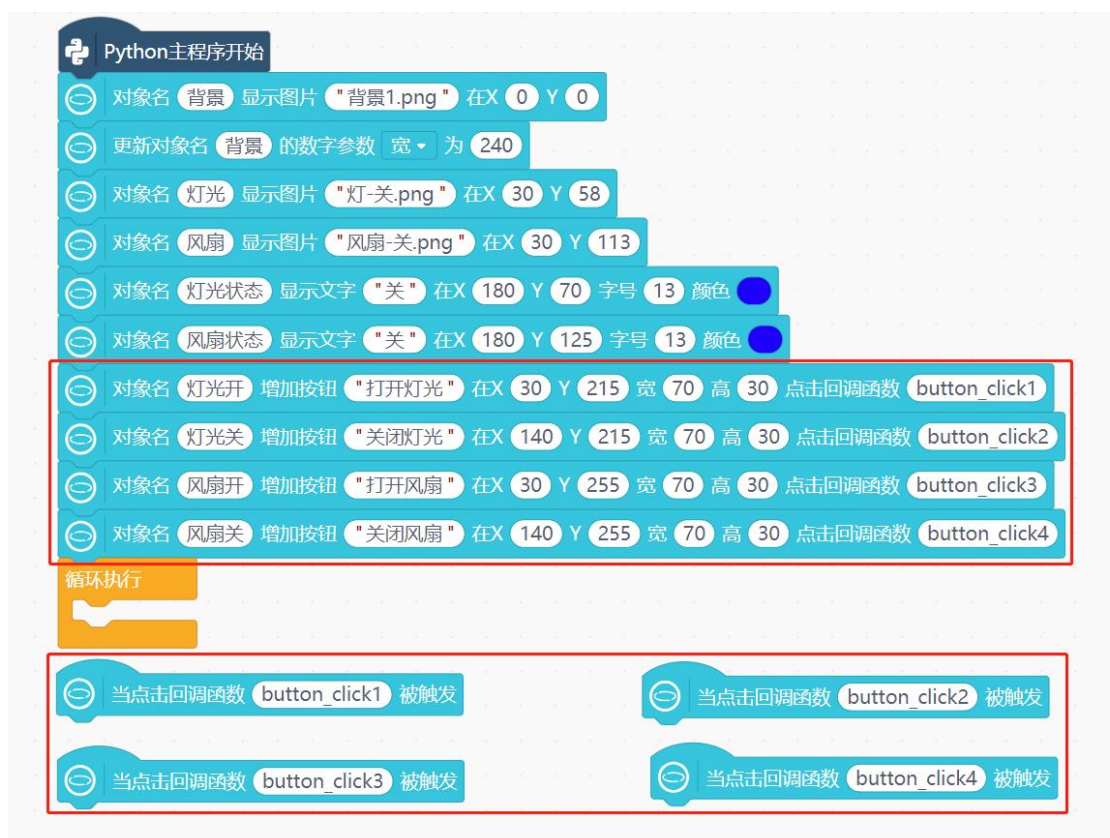
- (2) 使用**显示图片**指令，将文件系统中的“背景 1.png”、“灯-关.png”、“风扇-关.png”这些图片设置对应的XY坐标，显示在行空板上。然后使用**更新数字参数宽**指令，更新背景对象的宽为240。



- (3) 使用显示文字指令，设置灯光状态与风扇状态的提示文字为“关”，设置对应的xy坐标以及字号，显示在行空板上。



- (4) 使用对象名()增加按钮(“按钮”)在X0Y0宽()高()点击回调函数(button_click1)和当点击回调函数(button_click1)被触发指令，添加“打开灯光、关闭灯光、打开风扇、关闭风扇”这4个屏幕按钮。每个按钮的xy坐标如上图，并设置按钮的宽为70，高为30。

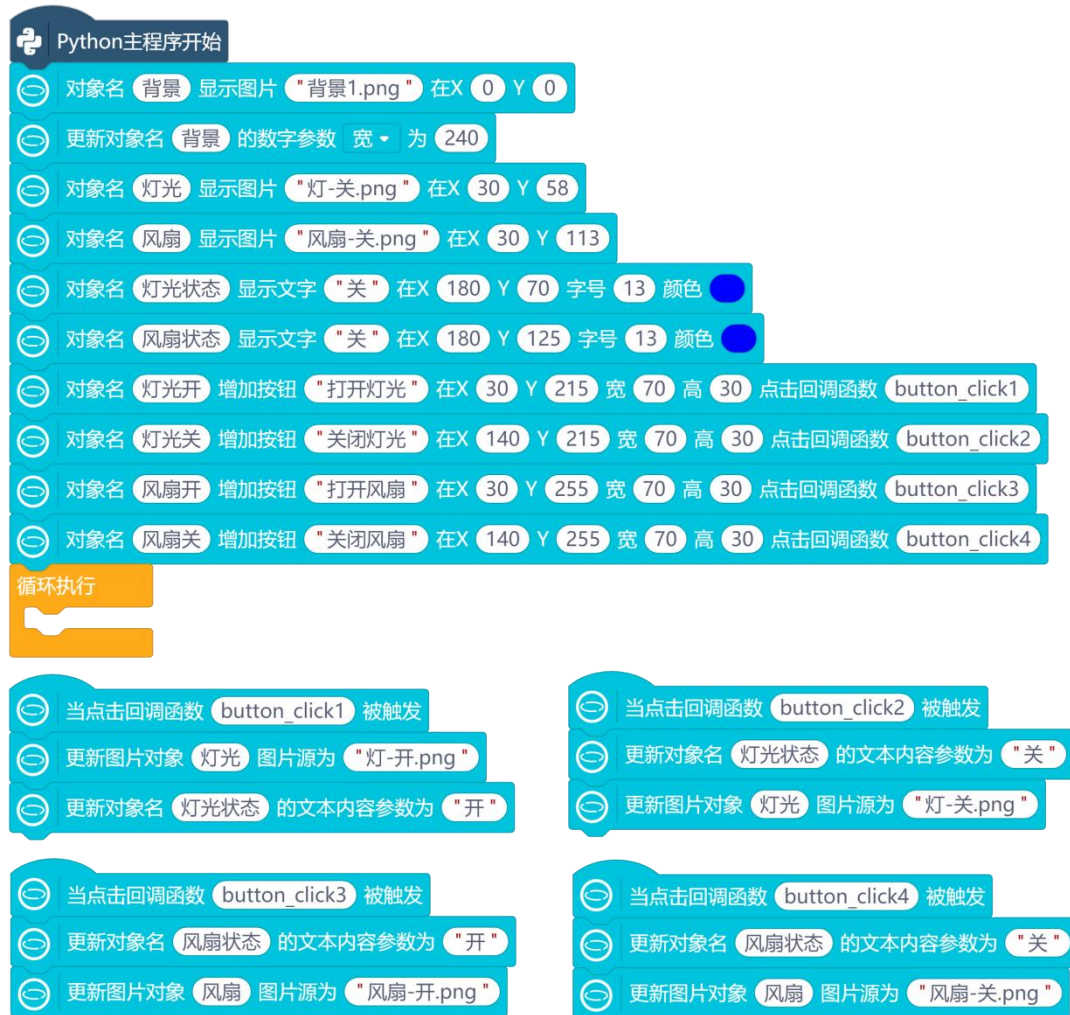


- (5) 当点击打开灯光按钮后，行空板上的灯光图片会切换为“灯-开.png”，灯光状态切换为“开”。在当点击回调函数（button_click1）被触发指令下，使用更新图片指令与更新文本内容参数指令，更新灯光对象的图片源为“灯-开.png”，更新灯光状态对象的文本内容参数为“开”。



接下来，大家以同样的方法，在当点击回调函数被触发指令下，更新其余三个按钮对应的对象图片源与文本内容参数。

4. 参考程序



6. 程序运行

检查行空板是否已连接到本地地址 10.1.2.3，检查无误后，点击右上角的运行按钮。程序运行成功后，点击行空板上对应的“打开/关闭”按钮，可以完成灯光和风扇图片以及对应状态的切换，效果如下。

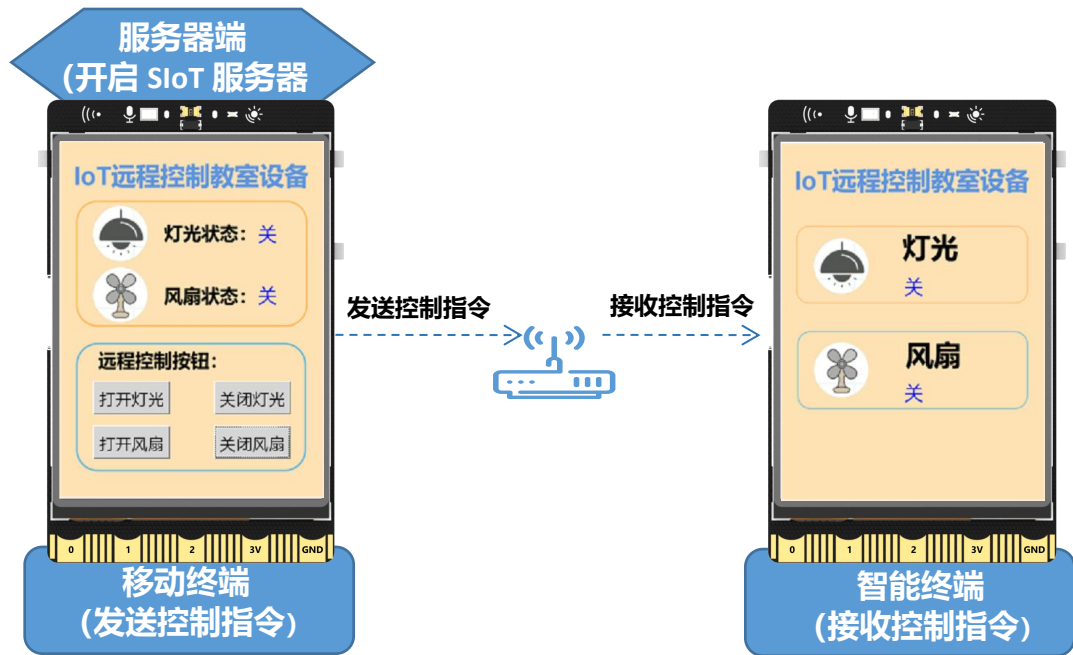


任务二：移动终端向 SIoT 发送控制指令

要实现移动终端（教师端）与智能终端（教室端）的数据传输，需要两块行空板来搭建物联网系统。移动终端利用行空板屏幕上的按钮，实现向 SIoT 物联网平台发送控制指令。说明移动终端也具备了服务器的发送和存储数据功能，那移动终端同样也是服务器端。而智能终端只需要接收指令后，根据指令控制设备执行对应的操作。

注意：物联网相关介绍，可以参考行空板图形化 Python 入门教程第 11 课

(<https://mc.dfrobot.com.cn/thread-314803-1-1.html>)

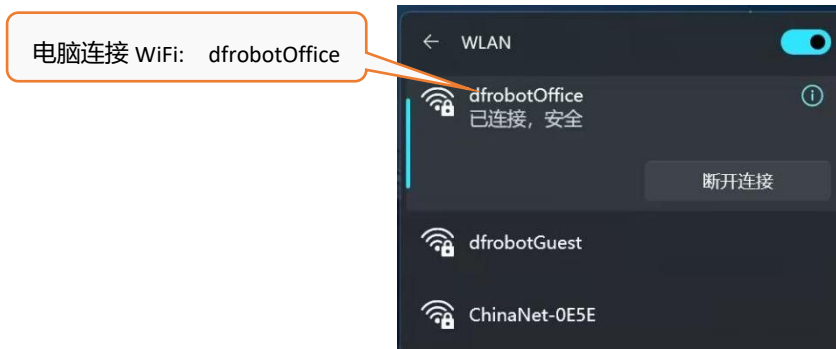


分析完两块行空板在物联网系统中的作用后，就需要做好行空板的网络配置，开启 SIoT 服务等准备工作，最后才能通过编程实现移动终端和智能终端间的数据传输。

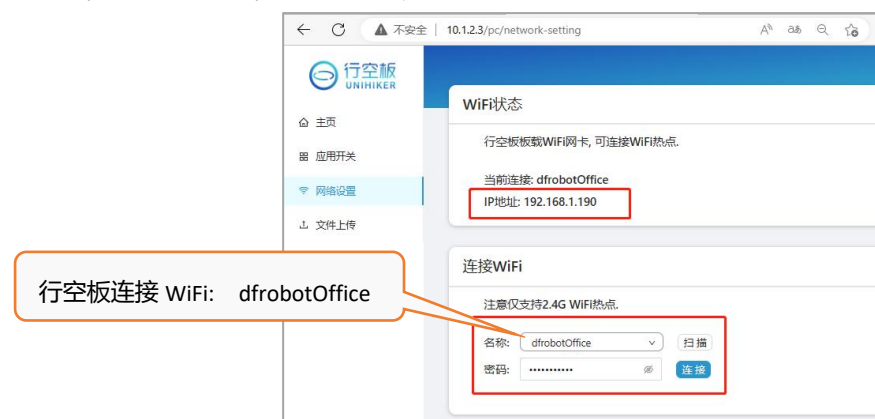
1. 准备工作

- (1) 按照任务一的方法，使用 USB 线将行空板连接到计算机，并准备好软件。
- (2) 配置网络：要实现移动终端与智能终端的数据传输，首先要将这两块行空板接入同一网络下，为了可

以通过电脑访问 SIoT 服务器, 电脑也需要接入同一网络下。首先, 将电脑连入路由器或手机的热点 (需要使用 2.4G WiFi) 。

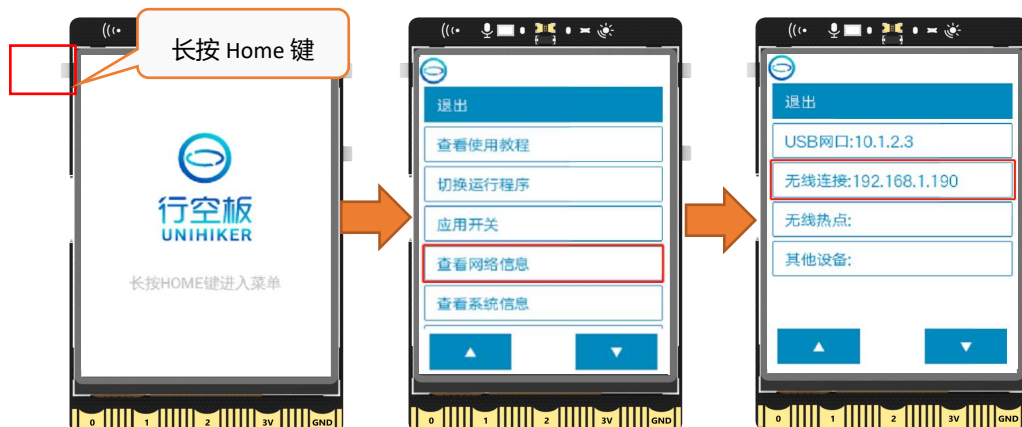


然后, 取一块行空板用 USB 线连接电脑, 并打开浏览器在地址栏输入 “10.1.2.3”, 进入行空板服务页面。接下来, 点击 “网络设置”, 在 “WiFi 连接” 中点击 “扫描” 寻找电脑连接的无线网, 输入正确的 WiFi 密码后, 点击 “连接”, 等待连接成功。



完成后, 我们用同样的方式给另一块行空板配置网络。最后, 需要给两块行空板同时上电, 使它们都保持开机状态。接下来, 我们需要分别进入两块行空板的 Home 菜单, 查看网络信息中的 IP 地址。

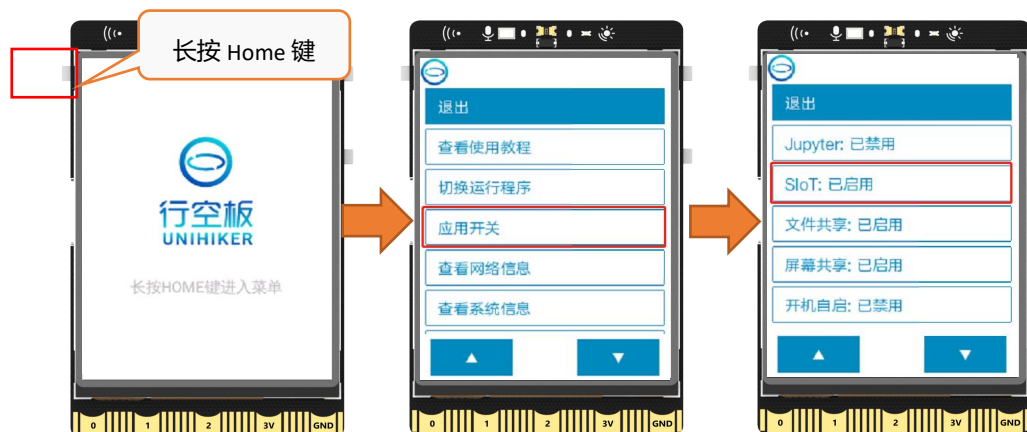
注意: 给行空板供电有三种方式: 1. 电源适配器 (手机充电头连接 type-c 口输出的 USB 线)。2. 充电宝 (充电宝连接 type-c 口输出的 USB 线)。3. 电脑 USB 端口 (通过 type-c 线将行空板和电脑连接)。



如果两块行空板的 IP 地址, 前三段的数字相同, 说明行空板在同一局域网下。现在, 选择其中一块行空板同时作为移动终端和服务端, 去开启 SIoT 服务器。这里, 我选择 IP 地址为: 192.168.1.190 这块行空板开启 SIoT 服务, 作为移动终端和服务端, 另一块 IP 地址为: 192.168.1.208 的行空板作为智能终端。



- (3) 开启行空板 SIoT 服务：长按行空板的“Home”键进入菜单页，点击“应用开关”，检查 SIoT 服务是否开启。如果显示“已禁用”，点击 SIoT 选项，切换 SIoT 状态为“已启用”。



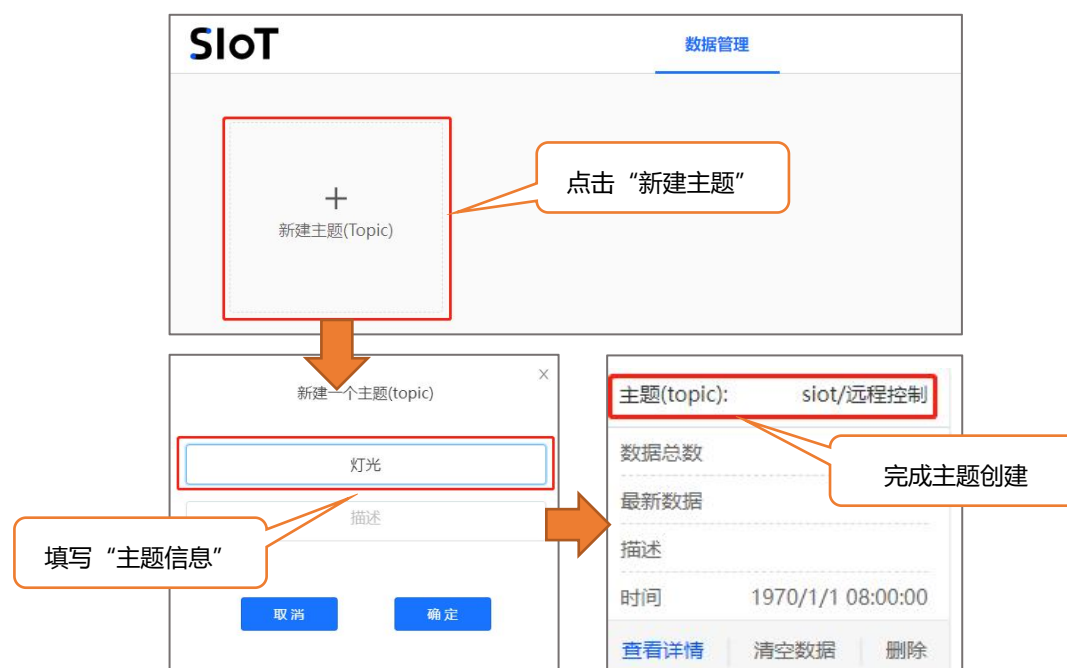
- (4) 进入 SIoT2.0 物联网平台：浏览器地址栏输入服务器端 IP 地址：192.168.1.190，进入行空板设置界面，进入应用开关，然后点击“打开页面”，进入 SIoT2.0 服务平台登录页面。输入对应的账号和密码，即可完成登录。**账号：**siot，**密码：**dfrobot。

注意：如果进入的不是 SIoT 2.0，可以按照知识园地中的方法对行空板 SIoT 服务器进行升级。



- (5) 创建主题：登录成功后，进入 SIoT 数据管理页面。然后点击“新建主题”，在弹出窗口中填写“主题名称”，点击“确定”完成主题的创建，页面会显示对应的主题标签。

注意：如果打开的还是 SIoT1.0 页面，可以使用 Ctrl+F5 强制刷新页面缓存。



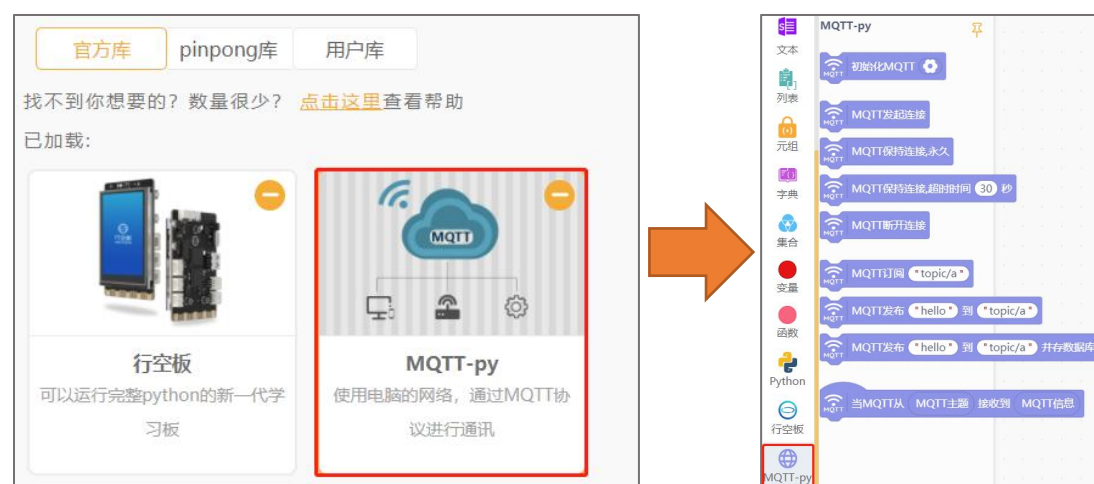
2. 编写程序

准备工作已完成，接下来就可以编程实现移动终端向 SIoT 发送控制指令了。因此，这个任务的功能在任务一程序的基础上进行添加。

MQTT 初始化

- (1) 要向 SIoT 平台发送数据，需要使用功能 MQTT 相关指令才能实现。但是现在行空板的指令区中没有这样的指令，该怎么办呢？点击“扩展”，在“官方库”中找到“MQTT-py”并点击，完成添加。

注意：“MQTT 协议”相关介绍，见知识园地。



- (2) 指令添加完成后，要使用 MQTT 构建物联网，首先需要在 **Python 主程序开始** 下，使用 **指令初始化 MQTT**。具体使用方法如下：点击指令的“设置”图标，将“SIoT 服务器”的内容修改为服务器端的行空板 IP 地址“192.168.1.190”。

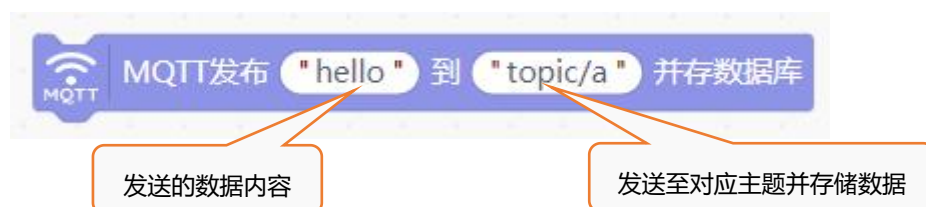


- (3) 初始化设置完成后, 使用 **MQTT 发起连接** 指令, 发起 MQTT 连接。连接成功后, 使用 **MQTT 保持连接,永久** 指令, 一直保持 MQTT 的连接。



移动终端向 SIOT 平台发送数据

- (1) 完成了“**MQTT 初始化**—**发起连接**—**保持连接**”这三个步骤后, 就可以在**当点击回调函数被 (button_click1)** 触发指令下, 使用 **MQTT 发布并存储数据** 指令, 向 SIOT 平台发送数据, 并将数据存储在 SIOT 服务器对应的主题中。



- (2) Button_click1 按钮的功能是打开灯光, 因此在**当点击回调函数被 (button_click1)** 触发指令下, 使用 **MQTT 发布并存储数据** 指令, 向 SIOT 平台发送文本数据“打开灯光”, 并将数据存储在 SIOT 服务器对应的主题“siot/远程控制”中。

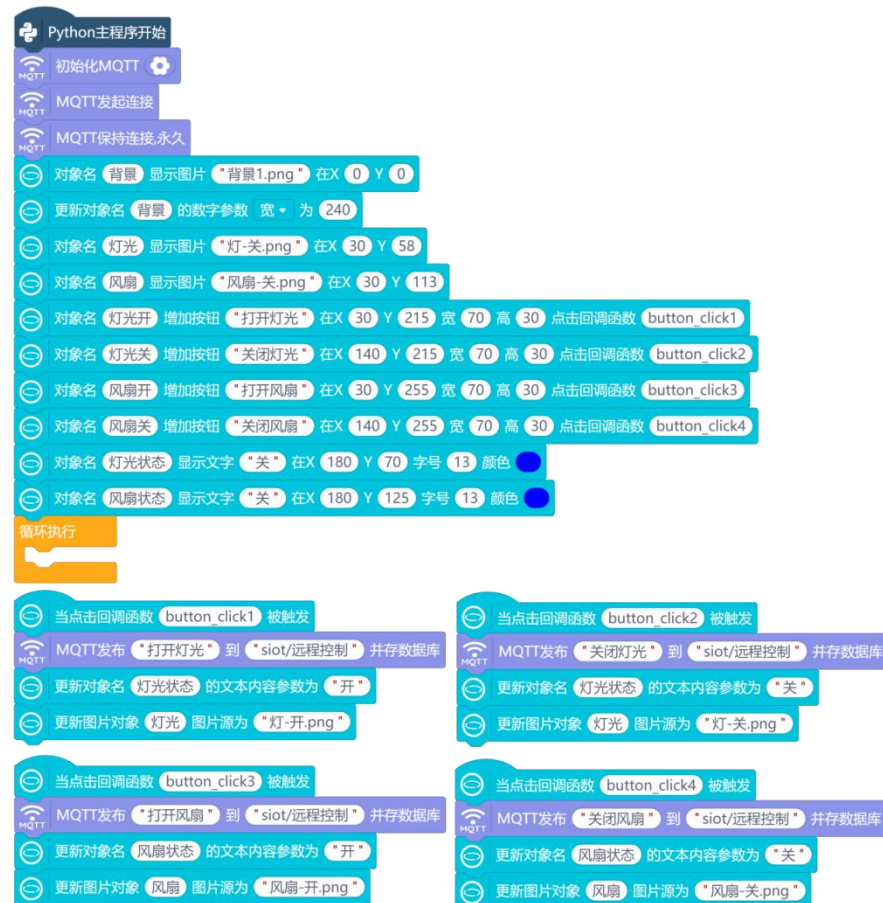


- (3) Button_click2 按钮的功能是关闭灯光，因此在当点击回调函数被 (button_click2) 触发指令下，使用 MQTT 发布并存储数据指令，向 SIoT 平台发送文本数据“关闭灯光”，并将数据存储在 SIoT 服务器对应的主题“siot/远程控制”中。



Button_click3、Button_click4 按钮的功能是打开、关闭风扇。接下来，大家按照上面的方式在当点击回调函数被 (button_click3) 触发指令与当点击回调函数被 (button_click4) 触发指令下，完成向主题“siot/远程控制”发送对应的“打开风扇”和“关闭风扇”文本数据。

3. 参考程序



4. 程序运行

这里我们需要在 Mind+ 中，通过行空板的 IP 地址实现远程连接终端。具体方法如下：在“连接远程终端”中，选择“手动输入”，然后输入服务器端的 IP 地址：192.168.1.190。



连接成功后，点击运行。程序运行成功后，打开 SiOT，选择主题 siot/远程控制，点击“查看详情”，进入主题的数据管理页面，然后勾选“自动刷新”功能。



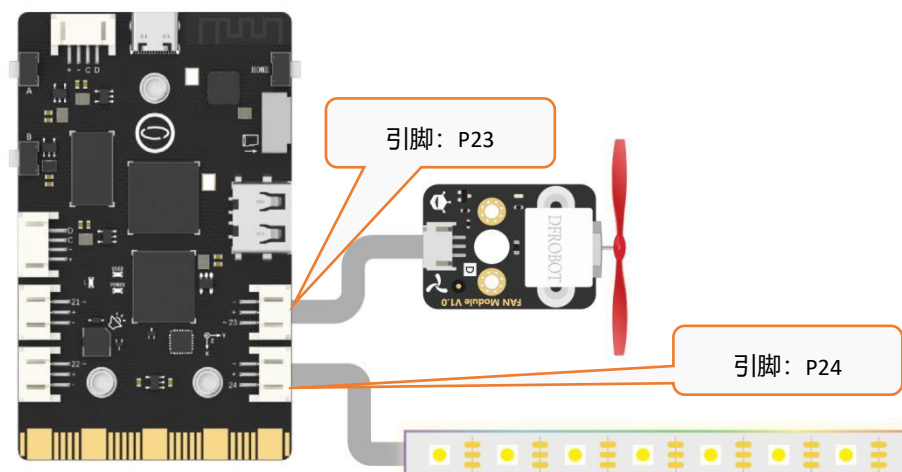
接下来，就可以通过行空板屏幕上的“打开灯光”、“打开灯光”按钮，向该主题发送“打开灯光”和“关闭灯光”文本信息。



任务三：制作智能终端

智能终端主要是接收移动终端发送的控制指令信息，然后根据对应的控制指令，控制连接在智能终端上的灯带以及风扇模块进行打开与关闭操作，同时切换界面的显示。因此，智能终端的主要功能分为三个：设计智能终端控制界面、智能终端接收控制指令、控制灯带与风扇设备。

1. 硬件连接



2. 准备工作

- (1) 打开 Mind+, 按照任务一完成软件的准备过程。
- (2) 添加 MQTT-py 库, 点击“扩展”, 在“官方库”中找到“MQTT-py”并点击, 完成添加。



- (3) 添加 WS2812 RGB 灯库, 点击“扩展”, 在 pinpong 库中找到“WS2812 RGB 灯”并点击, 完成添加。



3. 编写程序

设计智能终端控制界面

开始编写程序之前，先来分析一下智能终端的控制界面包含了哪些元素？背景、灯与风扇图片，以及设备状态。



- (1) 设计移动终端控制界面，为了简化程序，将不用改变的文字和背景框都制作成了背景图片。接下来，将图片素材库中的所有图片，都导入文件系统中。



- (2) 使用**显示图片**指令，将文件系统中的“背景.2.png”、灯-关.png”、“风扇-关.png”这些图片设置对应的xy坐标，显示在行空板上。然后使用**更新数字参数宽**指令，更新背景对象的宽为240。



- (3) 使用**显示文字**指令，将灯光状态与风扇状态的文字提示“关”，设置对应的 xY 坐标以及字号，显示在行空板上。



智能终端接收控制指令

- (1) 智能终端要接收服务器端的控制指令，所以也需要使用 MQTT 构建物联网。在 Python 主程序开始之下，完成“MQTT 的初始化——发起连接——保持连接”这三个步骤后，需要将 SIoT 服务器中的 IP 地址修改为：192.168.1.190。



- (2) 智能终端要接收服务器端存储的消息，要先订阅对应的主题才能接收消息。使用**MQTT 订阅**指令，订阅服务器端的主题“siot/远程控制”。



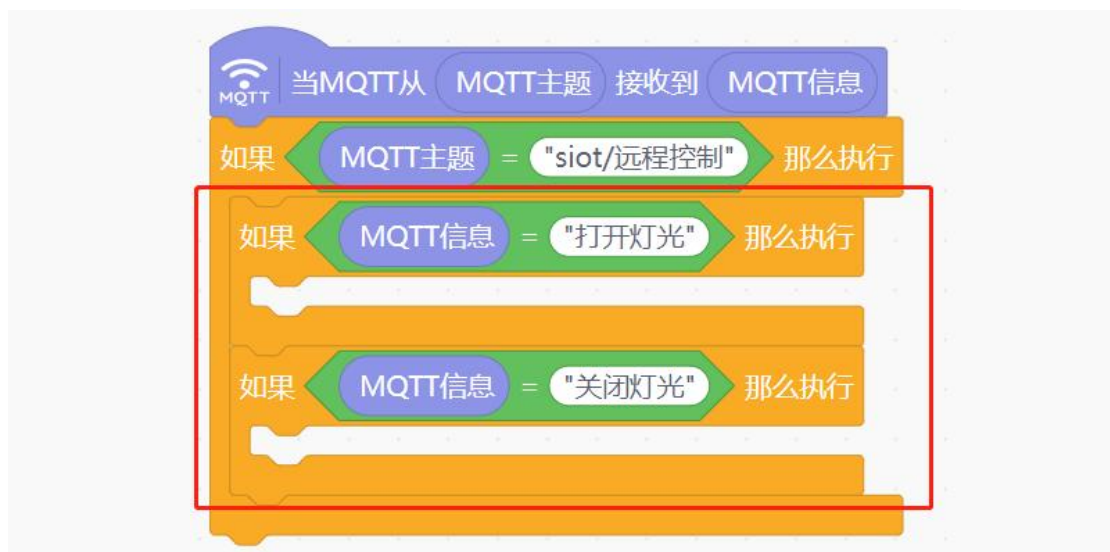
- (3) 使用**当 MQTT 从 MQTT 主题接收到 MQTT 信息**指令，来接收服务器端的控制指令。



- (4) 接收到数据后，首先需要判断数据来自的主题是否为“**siot/远程控制**”。因此，需要使用**条件判断**指令，对 **MQTT 主题** 进行判断。需要注意的是，**MQTT 主题** 的数据类型为字符串型，在进行 MQTT 主题判断时，需要将判断数据“siot/远程控制”放入英文状态下的引号内。



- (5) 同样使用**条件判断**指令，判断 **MQTT 信息** 的具体文本内容是“打开灯光”，还是“关闭灯光”。



控制灯带设备

- (1) 使用**RGB 灯初始化**指令，在 **Python 主程序开始**之下，初始化灯带连接到**行空板的引脚 P24**，并设置灯数为 7。



- (2) 当接收到的 MQTT 信息为“打开灯光”时，使用 **for 循环** 指令，控制 RGB 灯带显示蓝色；然后使用 **更新图片** 指令，更新灯光对象的图片源为“灯-开.png”；使用 **更新文本内容参数** 指令，更新灯光状态对象的文本内容为“开”。



- (3) 当接收到的 MQTT 信息为“关闭灯光”时，使用 **for 循环** 指令，控制 RGB 灯带显示黑色（关闭灯带）；然后使用 **更新图片** 指令，更新灯光对象的图片源为“灯-关.png”；使用 **更新文本内容参数** 指令，更新灯光状态对象的文本内容为“关”。

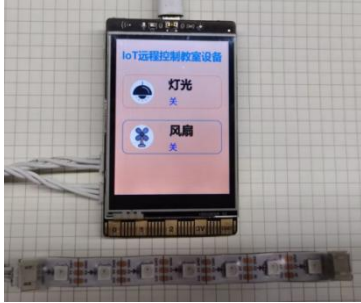


4. 参考程序



5. 程序运行

在“连接远程终端”中，选择“手动输入”，然后输入智能终端的 IP 地址：192.168.1.208。连接成功后，点击运行。接下来，就可以使用移动终端界面上的打开/关闭灯光按钮，控制智能终端上的灯带开启和关闭。

	
点击移动终端界面中的“打开灯光”按钮	智能终端接收到“打开灯光”指令后，打开灯光。
	
点击移动终端界面中的“关闭灯光”按钮	智能终端接收到“关闭灯光”指令后，关闭灯光。

6. 试一试

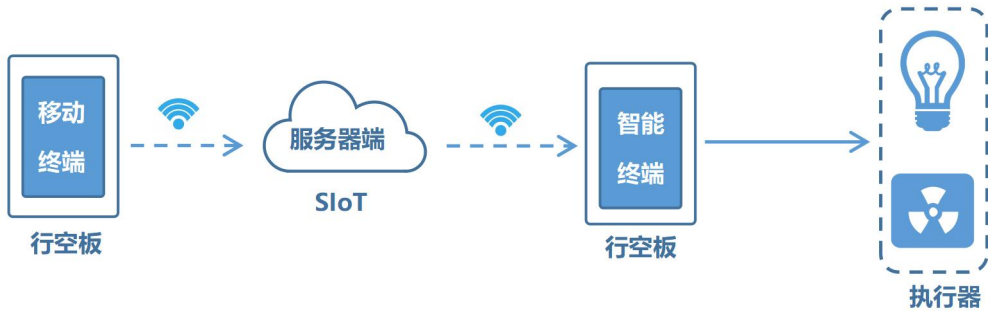
任务三的程序，只实现了控制 RGB 灯带模块的开启和关闭。接下来，大家在任务三的基础上添加程序，实现当接收到了“打开风扇”和“关闭风扇”的远程指令后，控制风扇模块的开启和关闭。

知识园地

1. 了解物联网远程控制

什么是物联网远程控制？远程控制是建立在 wifi、蓝牙、5G 等无线通信的基础上，将智能设备进行连接，实现数据的读取、远距离传输和远程无线控制，实现对设备的全自动管理。

这节课中，使用了两块行空板组建了物联网，这个系统中包含了智能终端、服务器端、移动终端。智能终端的行空板连接了多种执行器，负责接收数据并根据数据对执行器进行控制；移动终端用来查看数据，并远程控制执行器；服务器端是移动终端开启的 SIoT 服务功能，负责存储和收发数据。



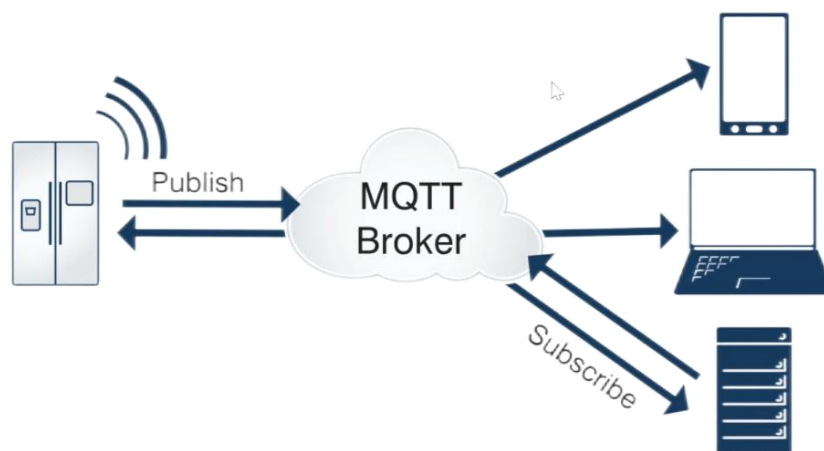
多行空板组建物联网的一般步骤为：配置网络，将所有物联网系统中的设备连入同一网段；开启服务器端 SIoT 服务；SIoT 连接设置，订阅主题，收发数据。当然，使用 SIoT2.0 时，要优先新建主题，才能成功订阅主题。



2. 了解什么是 MQTT 协议

MQTT (Message Queuing Telemetry Transport, 消息队列遥测传输协议) 是一个基于客户端 - 服务器的消息发布/订阅传输协议。

MQTT 协议在行空板与 SIoT 平台中起到一个什么作用呢？



我们可以将 MQTT 协议理解为行空板与 SIoT 平台通信的桥梁，在行空板和物联网 SIoT 平台通信过程中，MQTT 协议有三种身份：发布者(Publish)、代理(Broker)(服务器)、订阅者(Subscribe)。其中，消息的发布者和订阅者都是设备端（行空板），消息代理是服务器，消息发布者可以同时是订阅者。

换种简单的理解方式，我们可以将这个消息代理理解为一个群，如果你想在这个群里发消息，首先你得进入这个群，才能发送消息，而进群的方式就是 MQTT 协议。进群后，发消息这个人就是发布者，群就是服务器，其他群成员就是订阅者，你发的消息自己也能看到，因此发布者也可以是订阅者。

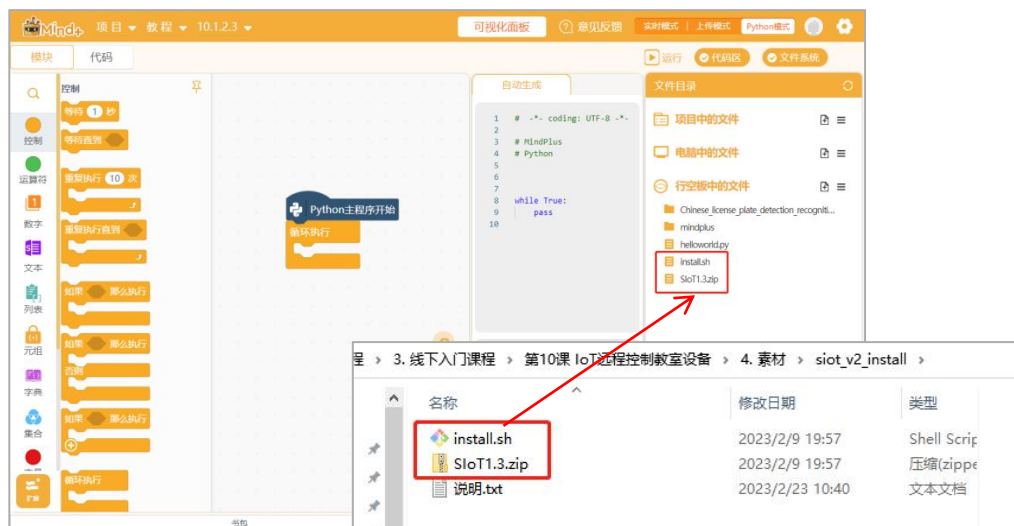
3. 行空板升级 SIoT 2.0 服务器

将行空板内置的 SIoT 1.0 替换为 SIoT 2.0，具体操作方法如下：

- (1) Mind+连接行空板（如果已连接，先断开连接再重新连接），然后打开文件系统。



(2) 将素材文件夹中的 install.sh 和 SioT1.3.zip 两个文件拖入“行空板中的文件”根目录里面。



(3) 在终端输入命令：bash install.sh，并回车。



(4) 当升级操作完成后，行空板会自动重启，SioT 2.0 服务器升级成功。两块行空板，都需要进行 SioT 2.0 的升级。

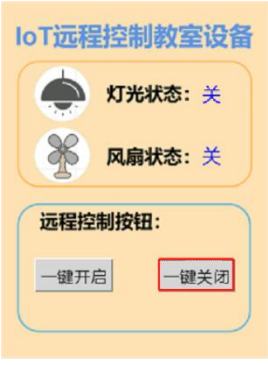
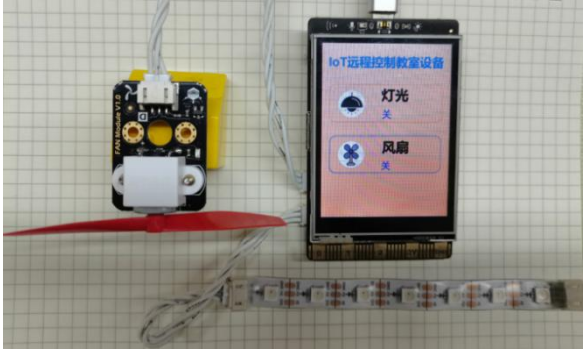
4. 指令学习

Block	功能说明
	该指令用于初始化设置 MQTT, 使用时需要点击设置图标, 选择 MQTT 平台, 写明服务器 IP、服务器端口号、用户名以及密码。
	该指令用于发起并建立 MQTT 连接。
	该指令用于一直保持 MQTT 连接。
	该指令用于订阅 MQTT 服务器的设备主题。主题名 (topic) 的格式为“项目 ID/设备号”。
	该指令用于向 MQTT 服务器的对应主题发送数据, 并将数据存储到主题对应的数据库中。
	该指令用于接收 SIOT 物联网平台的数据, 指令中包含 MQTT 主题和 MQTT 信息。

挑战自我

大家尝试一下, 在任务二的程序上将“打开/关闭灯光”按钮和“打开/关闭风扇”这 4 个按钮换成“一键开启”和“一键关闭”两个按钮, 在任务三智能终端程序上, 增加“一键开启/关闭”的 MQTT 信息判断。按下移动终端上的“一键开启”按钮, 智能终端设备上的灯和风扇同时打开; 按下移动终端上的“一键关闭”按钮, 智能终端设备上的灯和风扇同时关闭。

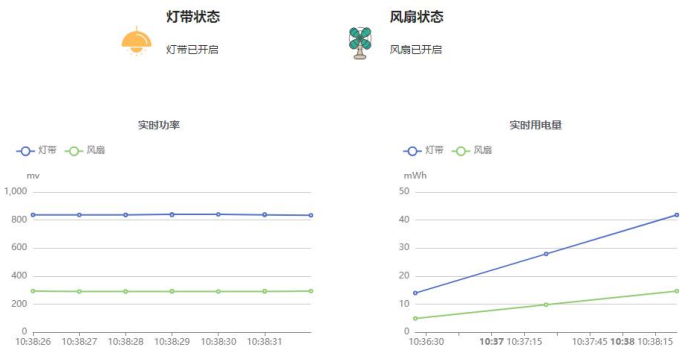
点击移动终端界面上的“一键开启”按钮	智能终端接收到“一键开启”指令后, 打开灯光、风扇。

	
点击移动终端界面上的“一键关闭”按钮	智能终端接收到“一键开启”指令后，关闭灯光、风扇。

第 11 课 IoT 教室设备电量统计

IoT 教室设备电量统计项目主要介绍如何使用物联网技术来实现教室设备电量的统计和监控。我们将使用传感器和物联网平台，收集教室内各种设备的用电数据，并对这些数据进行分析 and 可视化。通过本课程的学习，您将学会利用行空板、功率计、灯带、风扇等传感器搭建物联网设备、使用 mind+可视化界面进行数据分析，以及如何在教室管理中应用物联网技术。

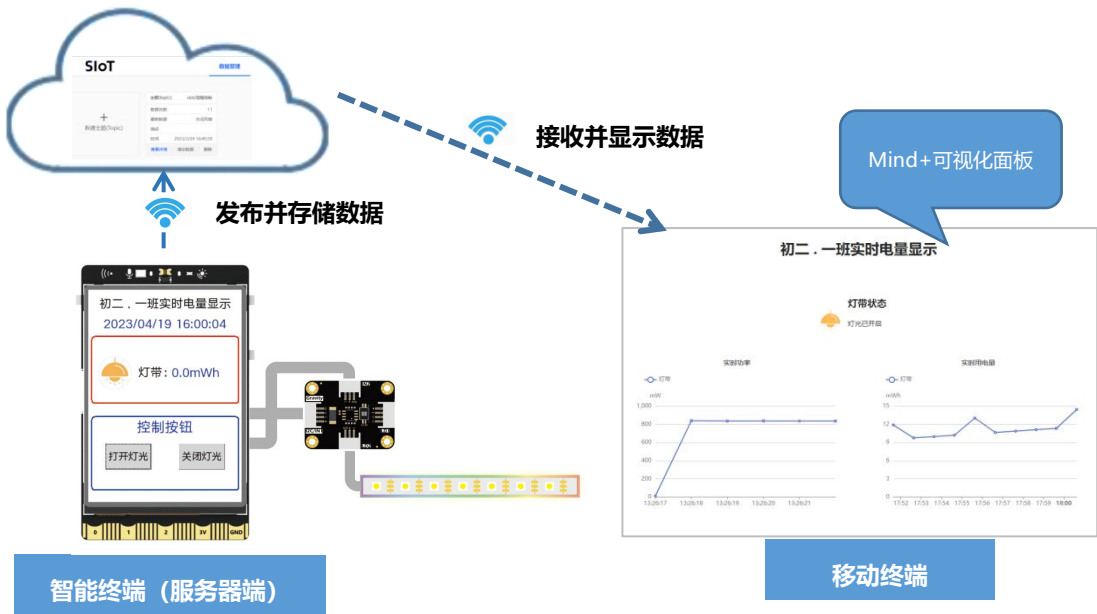
初二·一班实时电量显示



任务目标

IoT 教室设备电量统计系统主要由智能终端和移动终端组成。智能终端负责控制和实时采集设备的功率、用电量、状态等数据，并将这些数据传输到 SIoT 平台。服务器端则负责开启 SIoT 服务并存储数据。移动终端采用 Mind+可视化面板，可以远程显示设备的功率、用电情况和状态等信息，便于用户实时监测和管理设备。

注意：智能终端与服务器端使用的是同一块行空板。

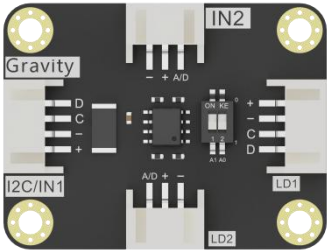


知识点

1. 掌握 I2C 数字功率计教育版的工作原理及使用方法。
2. 掌握 Mind+可视化面板的使用方法。
3. 了解物联网中的感知技术。
4. 了解 MQTT 协议中的 Topic。
5. 学习订阅多个 topic。

材料清单

硬件清单：

	
行空板 x1	RGB 全彩灯带 x1
	
I2C 数字功率计教育版 x1	Type-C USB 线 x1
	
白色硅胶绞线 x3	

软件使用：Mind+编程软件（1.8.0）

下载地址：<https://mindplus.cc/>

动手实践

IoT 教室设备电量统计系统主要由负责控制和实时采集数据的智能终端、存储数据的服务器以及远程查看数据的移动终端构成。为了更加直观地查看和分析数据，我们使用 Mind+可视化面板作为移动终端。下面，我们将通过四个简单的任务来学习如何制作 IoT 教室设备电量统计系统。

任务一：搭建智能终端设备

在这个任务中，我们将利用图片素材和按钮设计智能终端的控制界面。然后点击界面上的“打开灯光”或“关闭灯光”按钮控制灯带的开关状态。

任务二：采集灯带的用电情况

在这个任务中，我们将使用 I2C 数字功率计获取灯带的功率，并通过数学公式将功率转换为电量，最终在智能终端界面上显示电量信息。

任务三：搭建 IoT 教室设备电量统计系统

基于任务二的功能，我们将通过 wifi 把获取到的数据信息发布到 SIoT 物联网平台上。

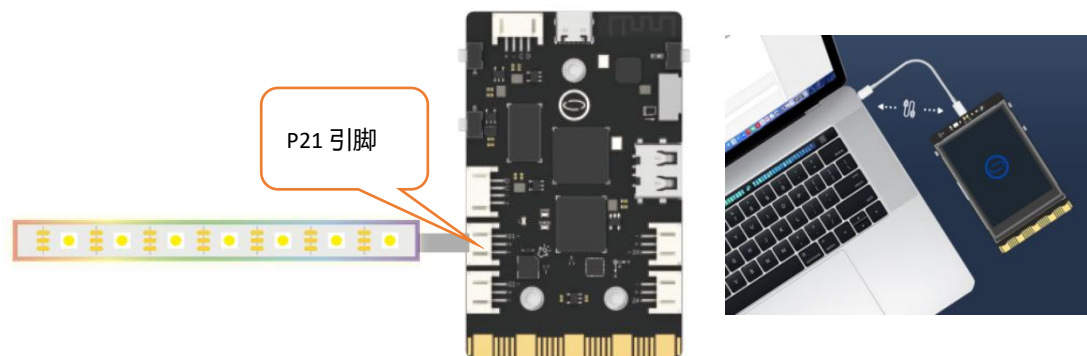
任务四：设计可视化面板

在这个任务中，我们将学习如何使用 Mind+数据可视化面板，并设计一个数据显示窗口，方便用户查看和分析数据。

任务一：搭建智能终端设备

1. 硬件连接

用一根 3pin 的连接线将灯带连接到行空板的 P21 引脚上，如下图。硬件连接成功后，使用 USB 线将行空板连接到计算机。



2. 软件准备

(1) 打开 Mind+，按照下面图示完成软件准备过程。



(2) 添加 WS2812 RGB 灯库，点击“扩展”，在 pinpong 库中找到“WS2812 RGB 灯”并点击，完成添加。



3. 编写程序

设计智能终端控制界面

首先观察一下，智能终端控制界面上都包含了哪些元素？标题、灯的 icon、打开灯光和关闭灯光按钮、两个矩形框、时间等元素，每个元素的坐标如下图。



- (1) 在前面的几节课中，为了简化程序，我们将不需要更改的文字和背景框都制作成了背景图片。在本任务中，我们将利用行空板的 block 模块设计背景界面，并通过“显示文字”指令将标题和时间以基准点为中心显示在行空板上，并将它们的对象名设置为“标题”和“时间”。



- (2) 使用“显示圆角矩形”指令在坐标 (10,70) 和 (10,190) 的位置画出宽为 220、高为 100、圆角半径为 5、线宽为 2 的圆角矩形边框，并将它们的对象名设置为“状态外框”和“控制外框”。



- (3) 在步骤 (2) 中，我们已经绘制了状态区域。接下来，使用“显示图片”指令在 (20, 100) 的坐标位置上显示图片“灯-关.png”，并将其对象名设置为“灯”；使用“显示文字”指令在 (80, 110) 的坐标位置上显示“灯带:”，并将其对象名设置为“灯带标题”。

注意：需要提前将图片素材“灯-关.png”与“灯-开.png”导入文件系统中。



- (4) 完成了状态区域的显示效果后, 接下来我们将通过以下几个步骤来完成控制区域的显示效果: 首先, 使用“增加按钮”和“当点击回调函数”指令添加两个屏幕按钮: “打开灯光”和“关闭灯光”, 并将它们的宽度设置为 70, 高度设置为 40; 然后, 使用“显示文字”指令和“更新对象名基准点”指令在控制区域添加“控制按钮”标题; 最后, 根据分析图的坐标设置按钮和标题的 XY 坐标。



控制灯带设备

我们已经完成了控制界面的设计, 想利用屏幕上的按钮来控制灯带的开关就需要完成这几个功能: 当点击“打开灯光”按钮时, 行空板上的灯光图片将切换为“灯-开.png”, 灯带会亮白色; 当点击“关闭灯光”按钮时, 行空板上的灯光图片将切换为“灯-关.png”, 灯带将熄灭。

- (1) 根据功能描述, 我们需要创建一个变量来控制灯带的开关状态。因此, 可以创建一个名为“light_ctrl”的变量用来接收按钮的状态。例如, 当“打开灯光”按钮被按下就将变量 light_ctrl 赋值为 1, 表示开启灯带; 当“关闭灯光”按钮被按下就将变量 light_ctrl 赋值为 0, 表示关闭灯带。



- (2) 使用“RGB灯初始化”指令，在“Python主程序开始”之下，初始化灯带连接到“行空板的引脚 P21”，并设置灯数为 7。



- (3) 在循环执行里面判断当 'light_ctrl=1' 时，使用“for 循环”指令控制灯带显示白色，并将对象名为“灯”的图片源更新为“灯-开.png”。如果 'light_ctrl' 不为 1，则控制灯带显示黑色，并将对象名为“灯”的图片源更新为“灯-关.png”。



为了能在屏幕上实时更新时间，可以在循环执行里面将对象名为“时间”的文本参数更新为 `获取系统时间[年/月/日 时:分:秒]`，程序效果见参考程序。

4. 参考程序

Python主程序开始

对象名 标题 显示文字 "初二.一班实时电量显示" 在X (120) Y (20) 字号 (13) 颜色 (●)

更新对象名 标题 的基准点为 中心

对象名 时间 显示文字 获取系统时间 年/月/日 时:分:秒 在X (120) Y (50) 字号 (13) 颜色 (●)

更新对象名 时间 的基准点为 中心

对象名 状态外框 显示圆角矩形 在X (10) Y (70) 宽 (220) 高 (100) 圆角半径 (5) 线宽 (2) 边框颜色 (●)

对象名 控制外框 显示圆角矩形 在X (10) Y (190) 宽 (220) 高 (100) 圆角半径 (5) 线宽 (2) 边框颜色 (●)

对象名 灯 显示图片 "灯-关.png" 在X (20) Y (100)

对象名 灯带标题 显示文字 "灯带:" 在X (80) Y (110) 字号 (13) 颜色 (●)

对象名 控制按钮 显示文字 "控制按钮" 在X (120) Y (205) 字号 (15) 颜色 (●)

更新对象名 控制按钮 的基准点为 中心

对象名 开灯光 增加按钮 "打开灯光" 在X (30) Y (230) 宽 (70) 高 (40) 点击回调函数 light_on

对象名 关灯光 增加按钮 "关闭灯光" 在X (140) Y (230) 宽 (70) 高 (40) 点击回调函数 light_off

将变量 light_ctrl 赋值为 0

初始化 RGB灯 np1 引脚 行空板引脚 P21(~A) 灯数 (7)

循环执行

更新对象名 时间 的文本内容参数为 获取系统时间 年/月/日 时:分:秒

如果 变量 light_ctrl = 1 那么执行

使用 number 从范围 0 到 6 每隔 1

np1 设置第 变量 number 个灯的颜色 (●)

更新图片对象 灯 图片源为 "灯-开.png"

否则

使用 number 从范围 0 到 6 每隔 1

np1 设置第 变量 number 个灯的颜色 (●)

更新图片对象 灯 图片源为 "灯-关.png"

更新时间

当点击回调函数 light_on 被触发

全局global 变量 light_ctrl

将变量 light_ctrl 赋值为 1

当点击回调函数 light_off 被触发

全局global 变量 light_ctrl

将变量 light_ctrl 赋值为 0

5. 程序运行

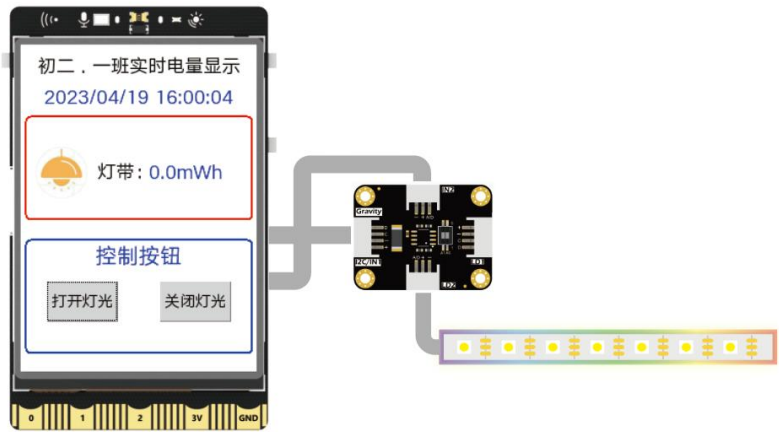
检查行空板是否已连接到本地地址 10.1.2.3, 检查无误后, 点击右上角的运行按钮。程序运行成功后, 点击行空板上对应的“打开/关闭”按钮, 可以开启/关闭灯带, 效果如下。

控制按钮	界面显示	设备状态
初始界面 (默认关闭灯带)	初二.一班实时电量显示 2023/04/19 18:37:25 灯带: 控制按钮 打开灯光 关闭灯光	

打开灯光	初二，一班实时电量显示 2023/04/19 18:38:57 灯带： 控制按钮 打开灯光 关闭灯光	
关闭灯光	初二，一班实时电量显示 2023/04/19 18:37:25 灯带： 控制按钮 打开灯光 关闭灯光	

任务二：采集灯带的用电情况

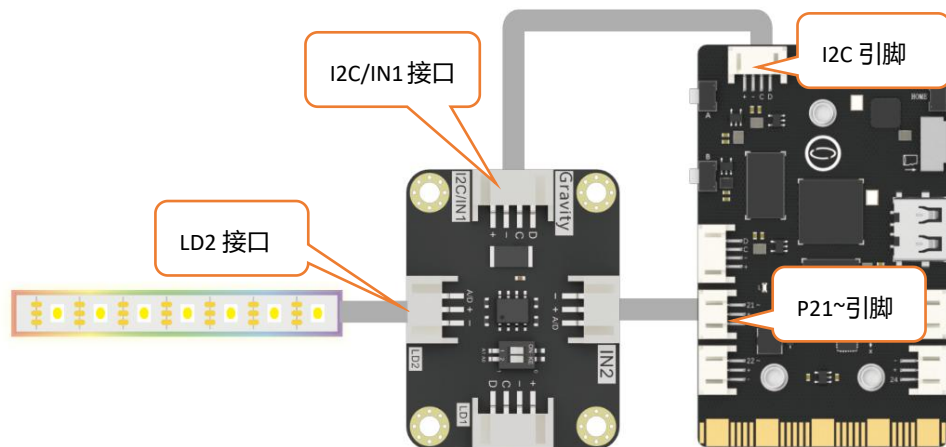
在任务一的基础上，使用 I2C 数字功率计获取灯带的功率，通过数学公式将功率转换为电量，并在智能终端界面上实时显示电量信息。



1. 硬件连接

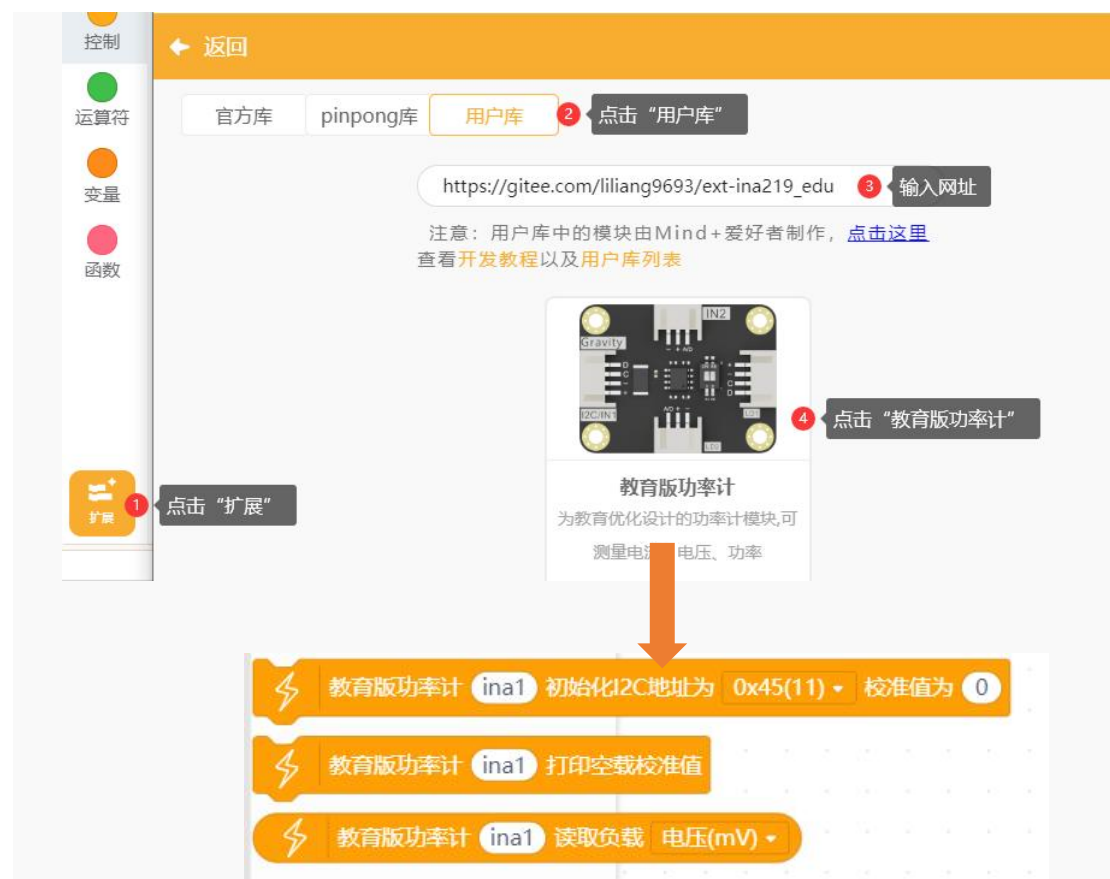
为了连接 I2C 数字功率计教育版到行空板上，需要使用一根 3pin 和一根 4pin 的连接线。其中，用 3pin 连接线将行空板的 P21 接口与功率计的 IN2 接口相连，用 4Pin 接口将行空板的 I2C 接口与功率计的 I2C/IN1 接口相连。接着，将灯带连接到数字功率计的 LD2 接口即可完成连接。如下图，硬件连接成功后，使用 USB 线将行空板连接到计算机。

注意：关于“[I2C 数字功率计教育版](#)”的相关介绍，见知识园地。



2. 软件准备

打开 Mind+软件，按照任务一的方法加载灯带库和行空板库。接下来，添加教育版功率计的库。点击“扩展”，在用户库的搜索框中输入地址“https://gitee.com/liliang9693/ext-ina219_edu”，然后按下回车键搜索“功率计”库。搜索完成后，点击“教育版功率计”库，完成添加。



3. 编写程序

准备工作完成后，基于任务一的程序，添加代码以使功率计能够读取灯带的功率。

初二，一班实时电量显示

2023/04/26 16:34:14



灯带用电量 (125, 110)

控制按钮

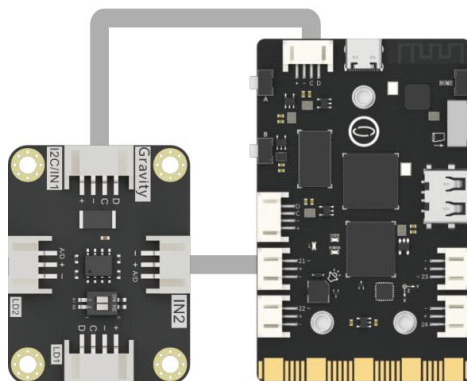
打开灯光

关闭灯光

- (1) 想要根据上图的位置显示灯带的用电量，可以使用“显示文字”指令在坐标 (125,110) 的位置显示一个字号为 12，颜色为蓝色的空值，并将它们的对象名设置为“灯带用电量”，方便后续调试。

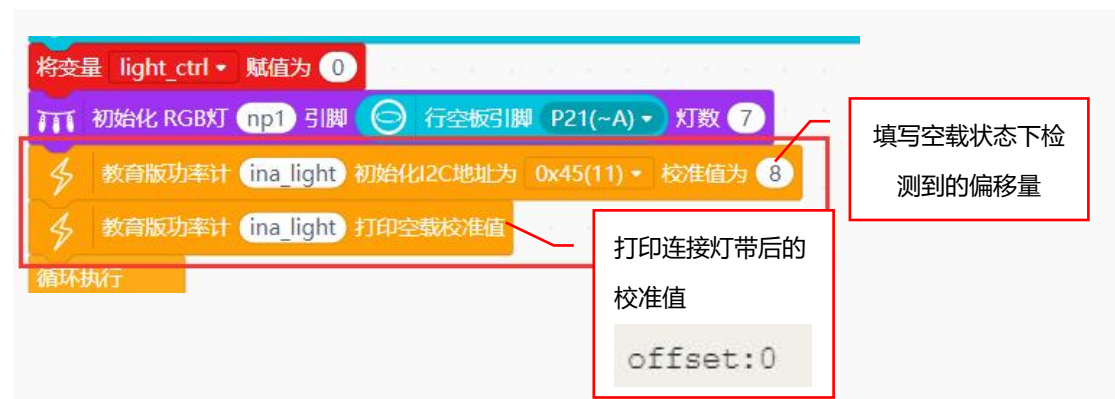


- (2) 在使用功率计之前，需要先进行校准。首先断开灯带与功率计的连接（电路连接如下图），然后运行下面的程序测量功率计的偏移量（offset）。



- (3) 现在已经知道该功率计的偏移量为 8。因此可以使用“教育版功率计 (ina1) 初始化 I2C 地址为 [0x45](11) 校准值为 (0)”指令校准偏移量，并将其名称设置为“ina_light”，这样可以方便后续调试。

注意：从步骤 (2) 开始，需要将灯带连接到功率计上，连接方式参考[任务二的硬件连接](#)。



- (4) 接下来，使用“教育版功率计 (ina1) 读取负载 [功率 mW]”指令获取灯带的实时输出功率，并将其名称设置为“ina_light”。然后创建一个变量“light_power”用来存储灯带的功率值。



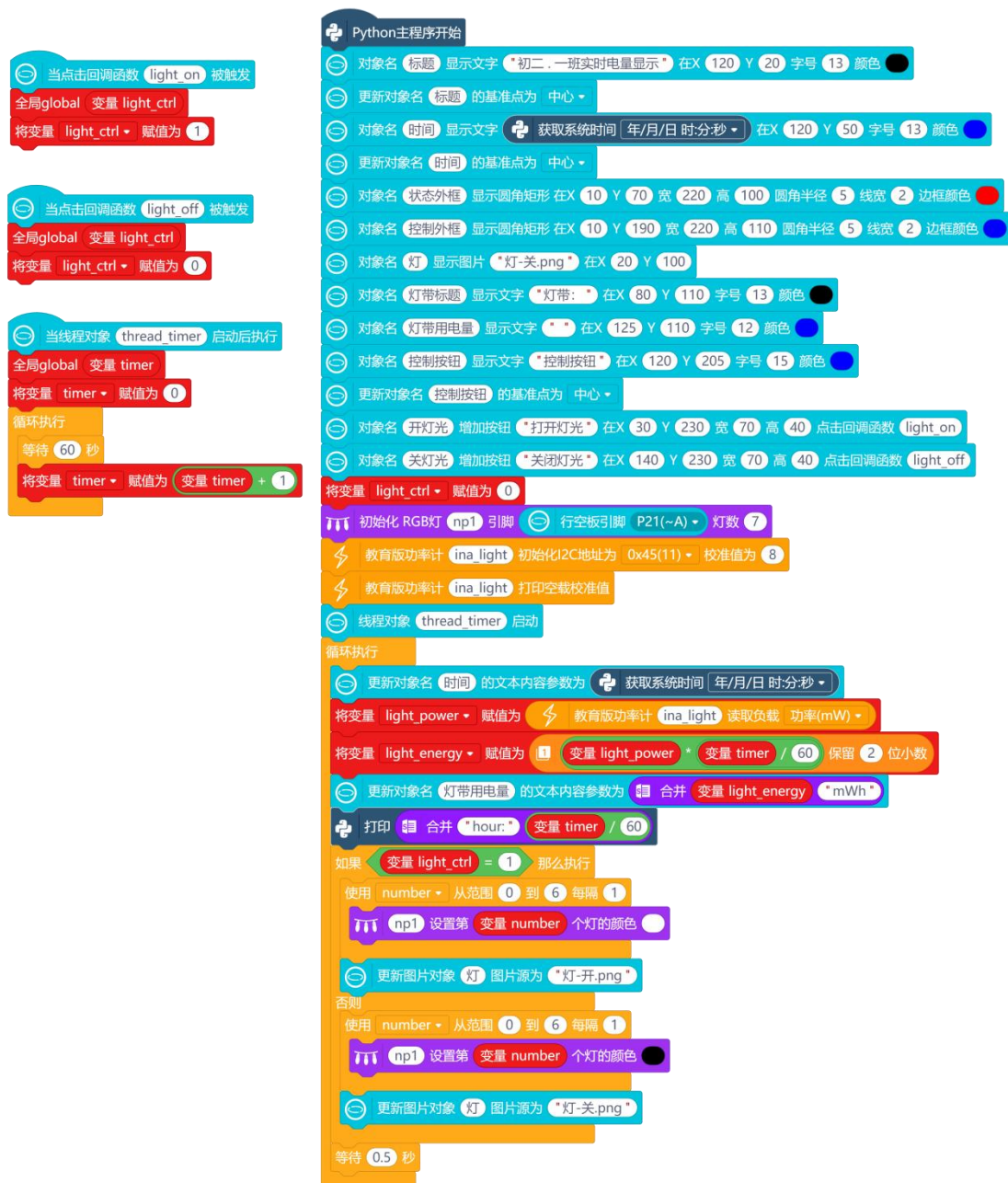
- (5) 根据前面的功能介绍，我们最终要实现的功能是采集灯带的用电量，所以可以使用公式“电量 (mWh) = 功率 (mW) * 使用时间 (h)”将功率转换成电量。首先新建变量“timer”用来计时。



因为 timer 的计时单位为分钟，所以需要使用公式“timer/60”将时间转换成小时，再利用功率转换成电量的公式计算灯带的用电量，并将其结果储存在变量“light_energy”中。同时使用“更新文本内容”指令更新对象名“灯带用电量”的内容。



4. 参考程序



5. 程序运行

检查行空板是否已连接到本地地址 10.1.2.3，检查无误后，点击右上角的运行按钮。程序运行成功后，观察行空板上是否有实时更新灯带的用电量？其效果如下。

控制按钮	打开灯光	关闭灯光
界面显示	初二，一班实时电量显示 2023/04/26 16:57:57 灯带:394.72mWh 控制按钮 打开灯光 关闭灯光	初二，一班实时电量显示 2023/04/26 16:58:44 灯带:6.57mWh 控制按钮 打开灯光 关闭灯光

任务三：搭建 IoT 教室设备电量统计系统

由前面任务目标得知，智能终端负责控制和实时采集数据，并将这些数据传输到 SIoT 平台。服务器端则负责开启 SIoT 服务并存储数据。Mind+可视化面板，可以远程显示设备的数据信息。接下来，基于任务二的功能，我们将获取到的数据信息发布到 SIoT 物联网平台上，并在 Mind+可视化界面上设计一个数据显示窗口。

1. 硬件连接

根据任务二的“硬件连接”连接好行空板与传感器。

2. 软件准备

- 打开 Mind+，按照任务一和任务二完成行空板、灯带与教育版功率计软件库的基本准备过程。
- 添加 MQTT-py 库，点击“扩展”，在“官方库”中找到“MQTT-py”并点击，完成添加。



3. 编写程序

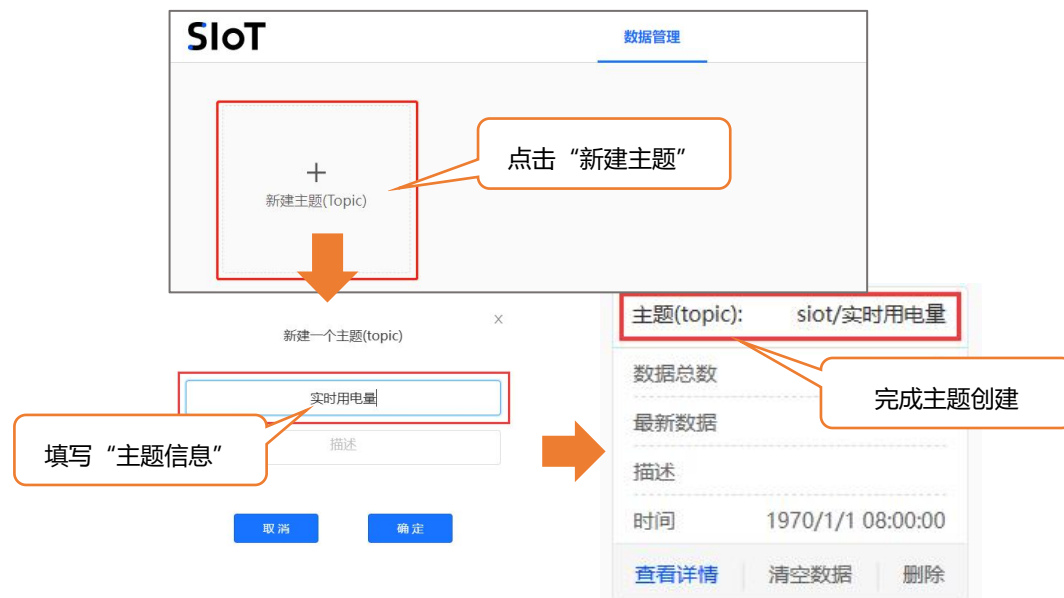
开始编写程序之前，先来分析一下智能终端需要发布哪些数据？灯带的实时用电量、实时功率与灯带的状态。所以我们需要进入 SIoT 物联网平台，并在 SIoT 数据管理页面里面新建 3 个主题。

- 进入 SIoT2.0 物联网平台：浏览器输入服务器端（智能终端）IP 地址：192.168.1.190，进入应用开关，然后点击“打开页面”，进入 SIoT2.0 服务平台登录页面。输入对应的账号和密码，即可完成登录。

账号：siot，密码：dfrobot。



- 创建主题：登录成功后，进入 SIoT 数据管理页面。然后点击“新建主题 (Topic)”，在弹出口中填写“主题名称”，点击“确定”完成主题的创建，页面会显示对应的主题标签。



按照上面的方法，完成“siot/实时用电量”、“siot/实时功率”、“siot/灯光状态”这三个主题的创建。

主题(topic):	siot/实时功率	主题(topic):	siot/实时用电量	主题(topic):	siot/灯光状态
数据总数	27514	数据总数	0	数据总数	1
最新数据	842.49	最新数据		最新数据	灯带已关闭
描述		描述		描述	
时间	2023/4/19 14:28:18	时间	1970/1/1 08:00:00	时间	2023/4/19 10:32:59
查看详情	清空数据	查看详情	清空数据	查看详情	清空数据
删除		删除		删除	

注意：开启 SIoT 服务的步骤为“配置网络——开启行空板 SIoT 服务——进入 SIoT2.0 物联网平台——创建主题”。其中配置网络和开启行空板 SIoT 服务的方法参考第 10 课 IoT 远程控制教室设备中的任务二。

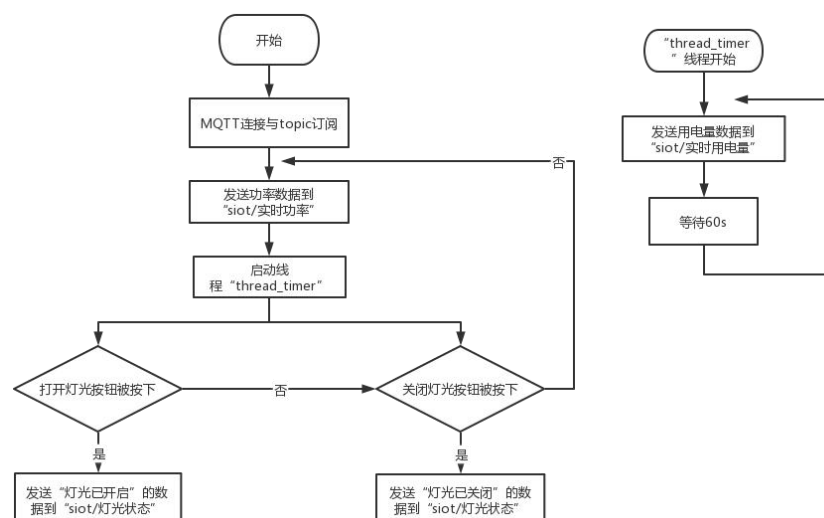
- (3) 想要向 SIoT 平台发送数据，需要使用 MQTT 相关指令构建物联网。首先在 Python 主程序开始之下，完成“MQTT 的初始化——发起连接——保持连接”这三个步骤，然后将 SIoT 服务器中的 IP 地址修改成服务器端（智能终端）的 IP 地址，我的行空板（服务器端）IP 地址为：192.168.1.190。



- (4) 智能终端要发布数据信息，要先订阅对应的主题才能接收消息。使用 MQTT 订阅指令订阅服务器端的主题“siot/实时用电量”、“siot/实时功率”、“siot/灯光状态”。



- (5) 完成了“MQTT 初始化——发起连接——保持连接——MQTT 订阅”这 4 个步骤后，我们需要进一步分析数据的发送时机，即实时用电量、实时功率与灯光状态的数据应该何时发送？



从上面的流程图可以看出，功率数据需要实时发送。因此，我们可以在循环执行的过程中使用“MQTT发布并存储数据”指令，将灯带的功率发送到“siot/实时功率”这个 topic 中，以实现实时监测。

The screenshot shows the MQTT publish block configuration: Topic is "siot/实时功率", Payload is the variable "light_power". Below it, a calculation block sets "light_energy" to $\text{light_power} \times \text{timer} / 60$ with 2 decimal places. The IoT platform interface shows the "siot/实时功率" topic with two data points:

时间	消息	操作
2023-4-27 13:38:37	10.2	删除
2023-4-27 13:38:37	17.0	删除

在前面我们已经成功地将灯带的功率发送到 Siot 物联网服务平台，现在我们需要将灯光状态也发送到平台上。根据流程图，当灯光按钮被按下时，我们就可以将“灯光已开启”的数据信息发送到“siot/灯光状态”这个 topic 中；否则，将“灯光已关闭”的数据信息发送到“siot/灯光状态”这个 topic 中。

The screenshot shows a conditional logic block: If "light_ctrl" equals 1, publish "灯光已开启" to "siot/灯光状态" and update the image to "灯-开.png". Otherwise, publish "灯光已关闭" to "siot/灯光状态" and update the image to "灯-关.png". A callout box states: "该指令不会将发送的数据保存到数据库。" (This instruction will not save the sent data to the database). The IoT platform interface shows the "siot/灯光状态" topic with a single data point: "灯光已开启".

现在，我们已经成功将实时功率和灯光状态的数据发送到了 Siot 物联网服务平台。接下来，需要在“thread_timer”线程中每 60 秒发送一次实时用电量数据。为了在线程中使用变量“light_energy”，我们需要将它定义为全局变量。然后，使用“MQTT 发布并存储数据”指令将“light_energy”的值发送到“siot/实时用电量”这个 topic 中。

The screenshot shows the "thread_timer" block configuration: Global variable "timer" is set to 0. Global variable "light_energy" is set to 0. The loop block publishes "light_energy" to "siot/实时用电量" and increments "timer" by 1 every 60 seconds.

4. 参考程序

当点击回调函数 **light_off** 被触发

全局global 变量 **light_ctrl**

将变量 **light_ctrl** 赋值为 **0**

当点击回调函数 **light_on** 被触发

全局global 变量 **light_ctrl**

将变量 **light_ctrl** 赋值为 **1**

Python主程序开始

初始化MQTT

MQTT发起连接

MQTT保持连接,永久

MQTT订阅 "siot/实时用电量"

MQTT订阅 "siot/实时功率"

MQTT订阅 "siot/灯光状态"

对象名 标题 显示文字 "初二一班实时电量显示" 在X 120 Y 20 字号 13 颜色 黑色

更新对象名 标题 的基准点为 中心

对象名 时间 显示文字 获取系统时间 年/月/日 时:分:秒 在X 120 Y 50 字号 13 颜色 蓝色

更新对象名 时间 的基准点为 中心

对象名 状态外框 显示圆角矩形 在X 10 Y 70 宽 220 高 100 圆角半径 5 线宽 2 边框颜色 红色

对象名 控制外框 显示圆角矩形 在X 10 Y 190 宽 220 高 110 圆角半径 5 线宽 2 边框颜色 蓝色

对象名 灯 显示图片 "灯-关.png" 在X 20 Y 100

对象名 灯光标题 显示文字 "灯带:" 在X 80 Y 110 字号 13 颜色 黑色

对象名 灯带用电量 显示文字 " " 在X 125 Y 110 字号 12 颜色 蓝色

对象名 控制按钮 显示文字 "控制按钮" 在X 120 Y 205 字号 15 颜色 蓝色

更新对象名 控制按钮 的基准点为 中心

对象名 开灯光 增加按钮 "打开灯光" 在X 30 Y 230 宽 70 高 40 点击回调函数 light_on

对象名 关灯光 增加按钮 "关闭灯光" 在X 140 Y 230 宽 70 高 40 点击回调函数 light_off

将变量 **light_ctrl** 赋值为 **0**

教育版功率计 ina_light 初始化I2C地址为 0x45(11) 校准值为 8

教育版功率计 ina_light 打印空载校准值

初始化 RGB灯 np1 引脚 行空板引脚 P21(~A) 灯数 7

线程对象 thread_timer 启动

循环执行

更新对象名 时间 的文本内容参数为 获取系统时间 年/月/日 时:分:秒

将变量 **light_power** 赋值为 教育版功率计 ina_light 读取负载 功率(mW)

MQTT发布 变量 **light_power** 到 "siot/实时功率" 并存数据库

将变量 **light_energy** 赋值为 变量 **light_power** * 变量 **timer** / 60 保留 2 位小数

更新对象名 灯带用电量 的文本内容参数为 合并 变量 **light_energy** "mWh"

打印 合并 "hour:" 变量 **timer** / 60

如果 变量 **light_ctrl** = 1 那么执行

使用 number 从范围 0 到 6 每隔 1

np1 设置第 变量 **number** 个灯的颜色 白色

MQTT发布 "灯光已开启" 到 "siot/灯光状态"

更新图片对象 灯 图片源为 "灯-开.png"

否则

使用 number 从范围 0 到 6 每隔 1

np1 设置第 变量 **number** 个灯的颜色 黑色

MQTT发布 "灯光已关闭" 到 "siot/灯光状态"

更新图片对象 灯 图片源为 "灯-关.png"

等待 0.5 秒

当线程对象 **thread_timer** 启动后执行

全局global 变量 **timer**

将变量 **timer** 赋值为 **0**

全局global 变量 **light_energy**

将变量 **light_energy** 赋值为 **0**

循环执行

MQTT发布 变量 **light_energy** 到 "siot/实时用电量" 并存数据库

等待 60 秒

将变量 **timer** 赋值为 变量 **timer** + 1

5. 程序运行

在“连接远程终端”中，选择“手动输入”，然后输入智能终端的 IP 地址：192.168.1.190。连接成功后，点击运行。程序运行成功后，在浏览器地址栏输入 IP 地址：192.168.1.190，查看灯带的实时功率，实时用电量及灯光状态。

实时功率

← 返回

当前主题: siot/实时功率

消息内容

发送

保存到数据库

开始时间

结束时间

100条

查询

导出数据

自动刷新

时间	消息	操作
2023-4-27 15:02:50	13.6	删除
2023-4-27 15:02:50	10.2	删除
2023-4-27 15:02:49	10.2	删除

实时用电量

← 返回

当前主题: siot/实时用电量

消息内容

发送

保存到数据库

开始时间

结束时间

100条

查询

导出数据

自动刷新

时间	消息	操作
2023-4-27 15:02:30	0.68	删除
2023-4-27 15:01:30	0.45	删除
2023-4-27 15:00:30	0.11	删除

灯光状态

← 返回

当前主题: siot/灯光状态

消息内容

发送

保存到数据库

开始时间

结束时间

100条

查询

导出数据

自动刷新

时间	消息	操作
----	----	----

暂无数据

注：灯光状态的数据没有设置保存在数据库中，所以进入 siot/灯光状态的 topic 里看不见数据。

6. 试一试

在整个项目中，我们一直采用手动控制灯带的方式。现在，我们尝试着修改程序，以实现自动控制灯带的亮灭。具体来说，需要在任务三的基础上修改一些逻辑判断，例如判断当前时间是否在 6 点到 21 点之间，判断环境光线是否小于 300 等。如果满足条件，则点亮灯带；否则，灯带将保持熄灭。

提示：需要使用“[获取系统时间](#)”指令。

任务四：设计可视化面板

1. 编写程序

IoT 教室设备电量统计系统已经基本搭建完了，为了让学生更方便地观察数据，可以使用 Mind+的“可视化面板”设计一个大屏显示器，连接物联网平台后它就是系统的可视化移动终端。

注意：有关“[Mind+可视化面板](#)”的区域介绍，见“知识园地”。

(1) 新建项目：点击 Mind+上方的“可视化面板”按钮，开启“Mind+数据可视化”窗口。



鼠标移至“新建项目”处，点击“新建空白项目”；在弹出窗口中填写好项目名称，点击“确定”项目管理页面就会出现新建项目标签。



(2) 连接 IoT: 新建好项目后，点击对应标签上的“编辑”，进入编辑页面，然后我们可以在页面上方的“菜单栏”，点击“未设置”，选择“数据源设置”，在弹出的窗口中填写要连接的物联网平台信息。“完成”后，显示订阅成功。



(3) 设计可视化界面：在可视化编辑页面中，先在右侧的“设置栏”设置画布尺寸、主题和封面。



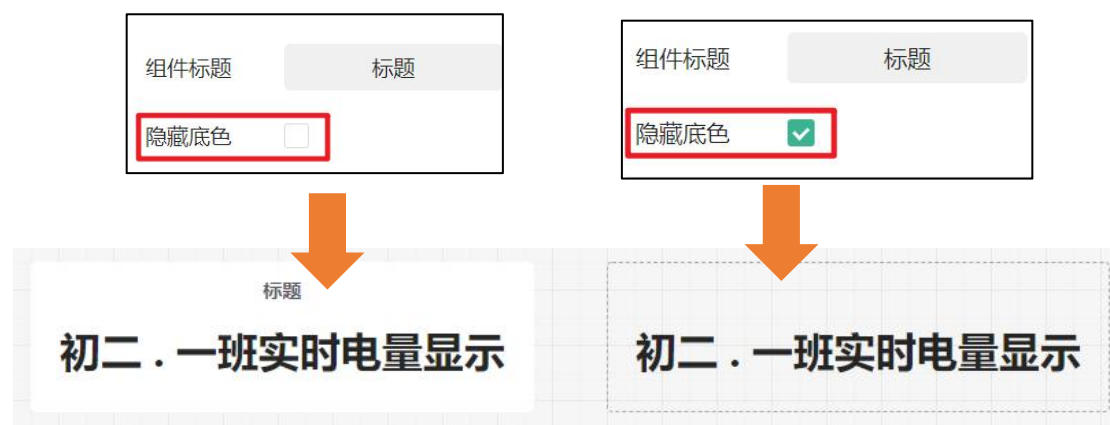
现在，我们就可以根据需求，在左侧的“组件区”寻找并添加组件，完成搭建和设计。接下来，我们将以下图示例中使用的组件为例，学习如何搭建和设置。



添加标题文字，首先在“组件区—装饰组件”分类下找到“文字”，像拖拽图形化指令一样拖出文字组件（删除拖回“组件区”即可）。然后，你可以拖动调整组件位置，在“设置栏”设置组件和显示字体的属性。



需要说明的是，这里的“组件底色”表示组件的外框，通常会显示组件标题。



根据上面的方法设置“灯带状态”的文字显示。



为了显示灯光状态，我们需要在 Mind+可视化面板上添加一个“单行文字”显示组件，并将其关联到对应的 Siot 主题。具体步骤为：首先在 Mind+可视化面板上，添加一个“单行文字”显示组件；然后找到“单行文字”组件的“组件属性”，点击“Topic”选项后，在右侧的下拉菜单中选择要关联的主题“siot/灯光状态”，选择完成后隐藏底色，数据即可自动显示在该组件中。



为了让灯光开启与关闭更加形象，我们还可以在 Mind+可视化面板上添加一个“自定义开关”基础组件，用来切换灯光状态的图片。例如：当主题“siot/灯光状态”接收到的信息是“灯光已开启”就显示灯亮的图片；如果接收到的信息为“灯光已关闭”就显示灯灭的图片。组件属性设置如下图。

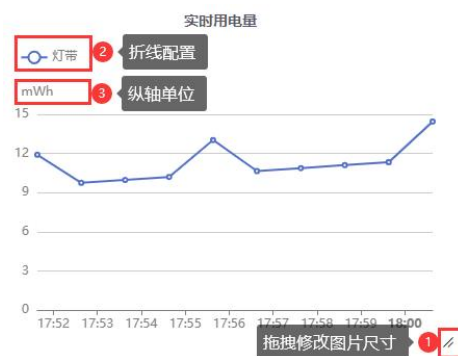
注意：“开/关的标志”消息需要和程序 MQTT 发布 “” 到主题并存储数据库 填写内容保持一致。



为了显示实时功率和实时用电量的数据，分析数据之间的规律。我们可以在 Mind+可视化面板上添加两个“折线图”图表组件，并将其关联到对应的 Siot 主题。以“实时功率”为例，首先在 Mind+可视化面板上，添加一个“折线图”图表组件；然后找到“折线图”组件的“组件属性”，将标题修改成“实时功率”；Topic 关联的主题为“siot/实时功率”；纵轴单位为“mW”；设置数量为“10”；折线配置修改成“灯带”。这些都设置完成后，数据即可自动显示在该组件中。



根据上面的方法设置“实时用电量”的图表显示。

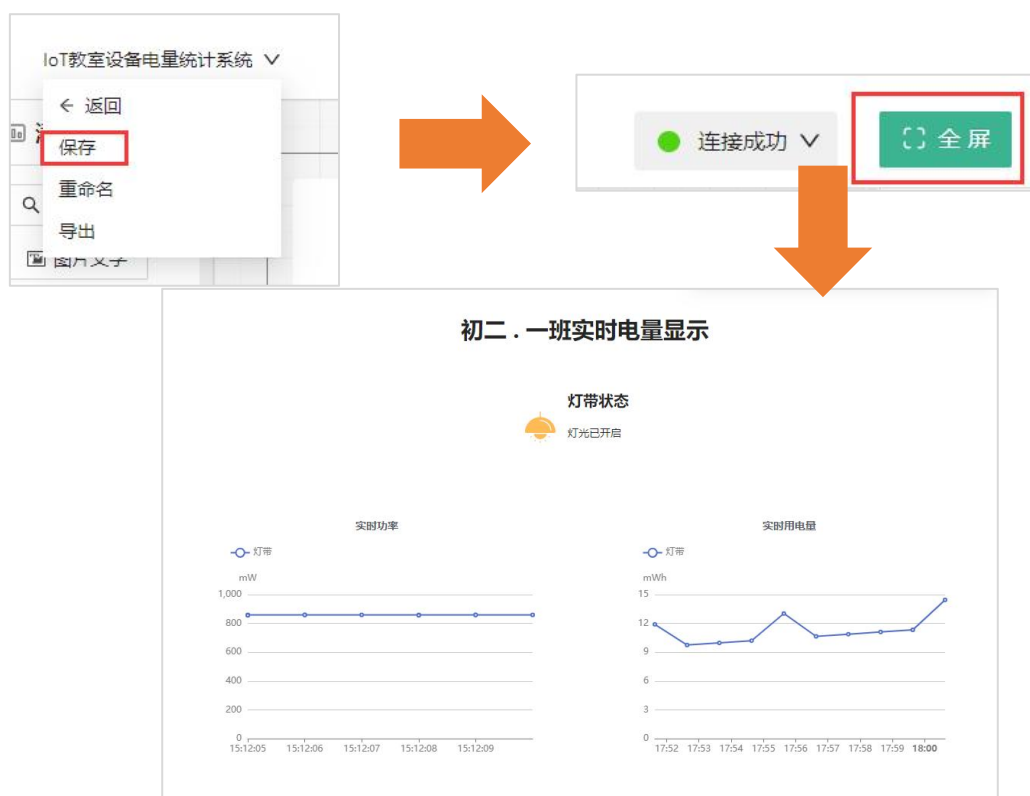


为了让界面更加美观，可以添加一个“色块”装饰组件，并将它移至最后层。然后在“组件属性”中将组件标题删掉；把组件宽度设置为“1100”，高度设置为“700”；组件颜色选白色。



2. 程序运行

点击项目名称下拉菜单中的“保存”，然后“全屏”查看整个可视化界面。



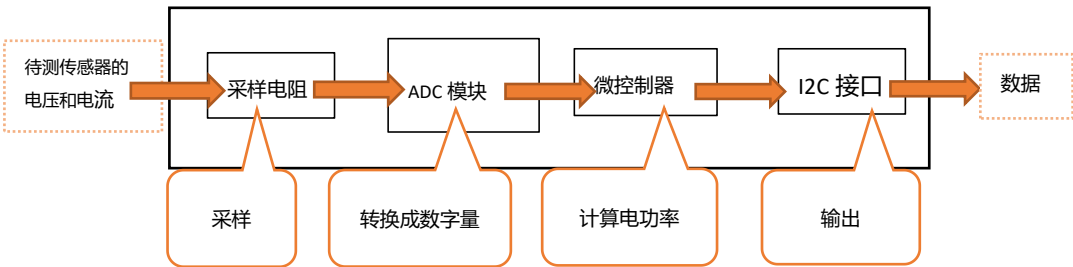
知识园地

1. 认识 I2C 数字功率计

这节课，我们学习了一种新型的传感器——I2C 数字功率计，它基于 I2C 总线协议，可用于测量各种传感器和执行器的电气参数，例如电压、电流、功率等。此外，该传感器通过 I2C 接口传输数据，以方便与其他电子设备进行通信和数据处理。I2C 数字功率计广泛应用于嵌入式系统、智能电源管理、电池管理和电力监测等领域，具有高精度、高可靠性和易于使用的特点。

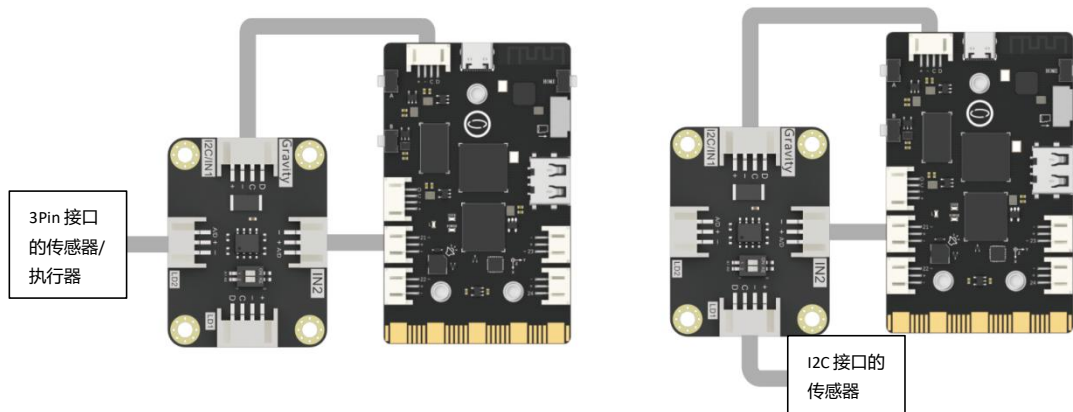
1) I2C 数字功率计工作原理

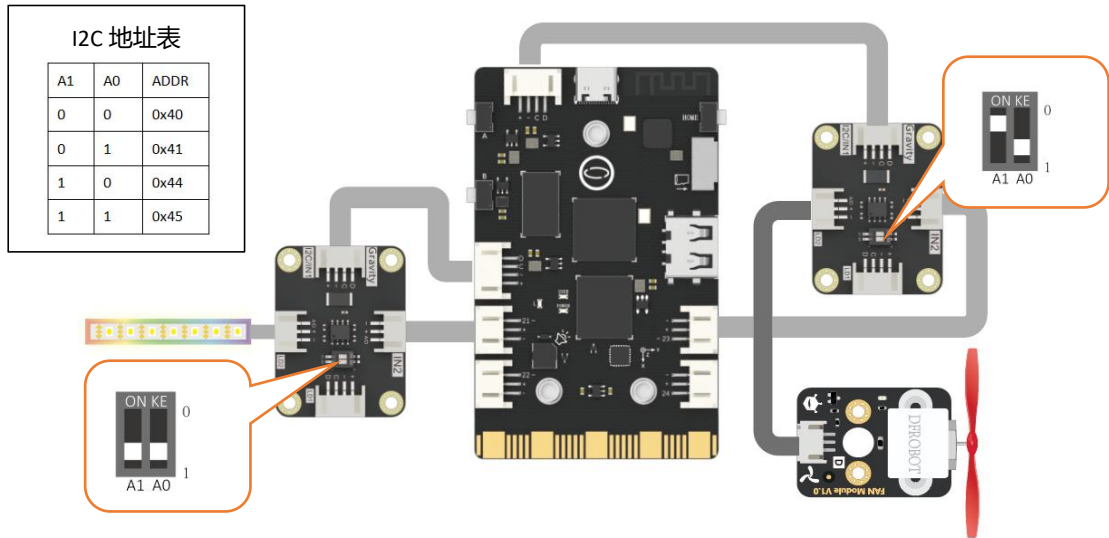
I2C 数字功率计的工作原理如下：首先，通过采样电阻将待测传感器的电压和电流引入到内部的 ADC（模数转换器）中进行数字化测量。接着，内部的 MCU 利用测量到的电压和电流值计算实时的电功率值。最后，测量到的电流、电压和功率值通过 I2C 接口输出到主控设备上，以方便其他电子设备进行通信和数据处理。



2) I2C 数字功率计的使用方法

I2C 数字功率计共有 4 个接口和 1 个拨码开关。其中，I2C/IN1 接口是用于 I2C 通信的接口，不仅可以功率计采集到的数据输出到主控设备，还可以通过它来控制 LD1 接口上的 I2C 传感器。IN2 接口是一个模拟/数字量通信接口，可用于控制 LD2 接口上的传感器/执行器。拨码开关用于切换 I2C 地址，以避免地址冲突。例如，如果要同时检测灯带和风扇的功率情况，则需要两块功率计模块，并将检测风扇的功率计 I2C 地址设置为 0x41(01)，检测灯带的功率计 I2C 地址设置为 0x45(11)，使用方法请参考下图。





2. 物联网中的感知技术

物联网是“万物之间通过连接互联网来共享信息并产生有用的信息，而且无需人为管理就能运行的机制”。而要实现这个机制，物联网中的感知技术就尤为重要。感知技术是指通过各种传感器和设备，对物理世界中的各种参数和事件进行感知和监测，从而获取数据并进行处理和分析。这些传感器可以检测物体的位置、温度、湿度、光强度、压力、声音、功率等物理量，同时也可以检测人体的生理指标，如心率、体温、血压等。以 IoT 教室设备电量统计系统为例，它的物联网原理示意图如下。



3. MQTT 协议中的 Topic

在 MQTT 协议中，Topic 是一种用于分类和归纳消息的标识符。Topic 是由一个或多个级别组成的，各级别之间使用斜杠“/”进行分隔。例如，一个 Topic 可以是“sensor/temperature/room1”，其中“sensor”是一级主题，而“temperature”和“room1”是二级主题。Topic 可以有任意多个级别，但每个级别的长度不能超过 64 个字符。在 SIoT 平台中，一级主题是固定的“siot”，我们只需设置二级主题的名称。

主题(topic):	siot/实时功率	主题(topic):	siot/实时用电量	主题(topic):	siot/灯光状态
数据总数	27514	数据总数	0	数据总数	1
最新数据	842.49	最新数据		最新数据	灯带已关闭
描述		描述		描述	
时间	2023/4/19 14:28:18	时间	1970/1/1 08:00:00	时间	2023/4/19 10:32:59
查看详情	清空数据	查看详情	清空数据	查看详情	清空数据
删除		删除		删除	

在 MQTT 中，发布者（Publisher）将消息发布到指定的 Topic 上，订阅者（Subscriber）则可以订阅一个或多个 Topic 以接收相应的消息。当发布者发布一条消息时，该消息将会被发送到与之关联的 Topic 上，而所有订阅了该 Topic 的订阅者都将会收到该消息。



Topic 在 MQTT 协议中起到了非常重要的作用，它可以用来对消息进行分类和过滤，从而使得订阅者只接收到与自己相关的消息。Topic 还支持通配符匹配，这使得订阅者可以更灵活地订阅多个相关的 Topic，以满足不同的业务需求。

4. Mind+可视化面板介绍

Mind+可视化面板是一款专为 IoT 设计的界面设计工具，用以更加直观和便捷的方式呈现接收到的数据。使用该工具，你可以通过拖拽组件来设计想要的数

据展示和控制界面。在项目中，我们已经学习了 Mind+ 可视化面板的组成和基本操作。现在，我们将深入探讨该界面中各个区域的具体功能



区域名称	功能介绍
组件区	包含四组组件，组件使用时直接拖拽至搭建区即可，只有可视化面板在编辑时才显示。 基础组件：包括常用的交互类组件，收/发 MQTT 数据。 显示组件：主要为各种显示效果的组件，接收 MQTT 发过来的数据。

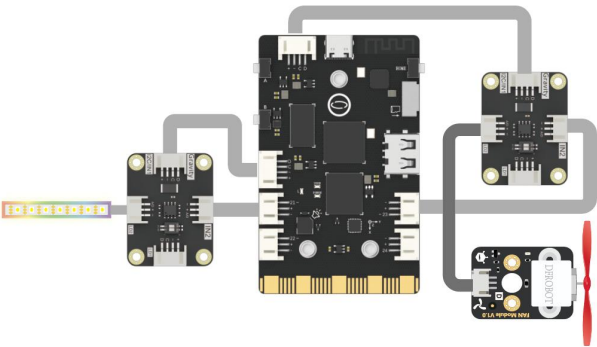
	图表组件：主要为各种图表类型的组件，接收 MQTT 发过来的数据。 装饰组件：无数据收发功能，主要是静态装饰作用，用来辅助界面设计。
搭建区	用于放置选择的组件，只有可视化面板在编辑时才显示。
设置栏	点击搭建区背景时，用于设置和显示画布尺寸；点击组件时，用于设置和显示组件尺寸、显示文字等。只有可视化面板在编辑时才显示。
菜单栏	包含项目名称操作、数据源设置、布局选择以及全屏/编辑按钮；全屏模式下，鼠标放至窗口上方才可显示，否则默认隐藏。

5. 指令学习

Block	功能说明
	该指令用于初始化设置功率计的 I2C 地址，并设置功率计的偏移量。
	该指令用于输出功率计的偏移量。
	该指令用于读取功率计检测到的电压、电流、功率等电气参数值。

挑战自我

大家尝试一下，在任务三的基础上，订阅一个 Topic 用来记录风扇的用电情况，并在可视化界面上显示风扇的状态、实时功率以及实时用电量。

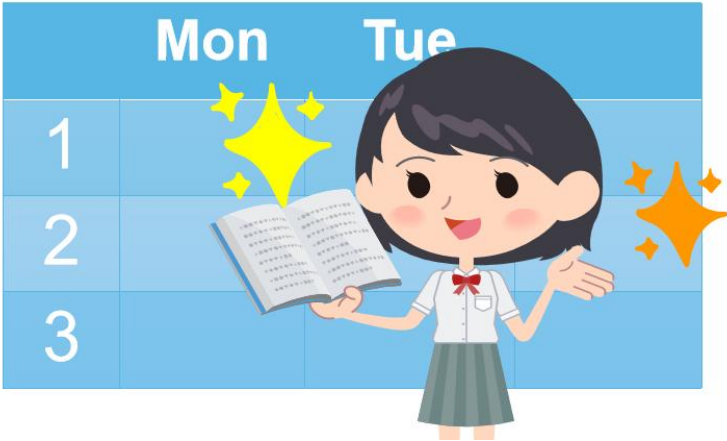


初二·一班实时电量显示



第 12 课 IoT 班级信息推送器

IoT 班级信息推送器，是一款较为实用的校园 APP，可以不仅帮助教务处老师轻松管理各班课程安排，还能让老师和学生便捷地查看自己的课程表，了解当天的课程信息以及上课时间。同时，班级端的智能终端设备还能实时显示该班级的应到人数和实到人数。接下来，就利用行空板和模拟量超声波传感器制作一款可以自动更新课程表信息的 IoT 班级信息推送器。



任务目标

移动终端从制定好的课程表中获取当天的课程信息，并将课程信息发送到智能终端（班级端）。智能终端接收课表信息后，将当天的课程安排显示在行空板上。同时，智能终端还能通过模拟量超声波传感器统计实到人数并将人数显示在行空板上。

		A	B	C	D	E	F	G
1	节次	星期	星期一	星期二	星期三	星期四	星期五	
2	上午	1	语文	数学	英语	语文	品德	
3		2	品德	英语	数学	品德	英语	
4		3	数学	品德	体育	数学	语文	
5		4	英语	语文	语文	英语	数学	
6	下午	5	语文	数学	品德	语文	美术	
7		6	劳动	美术	数学	数学	活动	

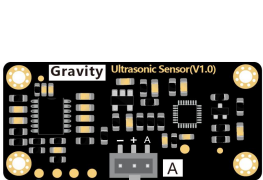
(工作表)

移动终端从工作表中
获取课程内容并传输。



(移动终端)

智能终端接收移动终
端的课程内容并显示。



智能终端通过超声波
传感器统计实到人数。



(智能终端)

知识点

1. 了解什么是智能终端和移动终端
2. 了解物联网常见的通信方式
3. 学习使用 xlrcl 库读取课表里的内容
4. 了解工作簿与工作表的区别
5. 了解超声波传感器的工作原理

材料清单

硬件清单：

	
行空板 x2	模拟量超声波测距传感器 x1
	
白色硅胶绞线 x1	Type-C USB 线 x2

软件使用：Mind+编程软件 x1

下载地址：<https://mindplus.cc/>

动手实践

移动终端（教务端）负责从制定好的课程表中获取当天对应的课程信息，通过 SIoT 平台将课程信息发送到智能终端（班级端）。智能终端接收课表信息后，将当天的课程安排显示在行空板上。同时，智能终端还能通过模拟量超声波传感器统计实到人数，并将实到人数和应到人数显示在行空板上。接下来，就从下面两个任务来完成 IoT 班级信息推送器的制作吧！

任务一：移动终端读取教务处制定的课表内容

首先，教务处需要制定高中课表，然后移动终端根据系统时间星期几去获取高中课表里面对应的课程安排，并上传到 SIoT 服务器端。

任务二：智能终端统计实到人数并显示获取的课表信息

智能终端从 SIoT 服务器端接收课程信息后，将当天的排课安排显示在行空板上。同时，智能终端还要通过超声波传感器进行人数统计，并显示应到人数和实到人数。

任务一：移动终端读取教务处指定的课程内容

1. 硬件连接

使用 USB 线将行空板连接到计算机。



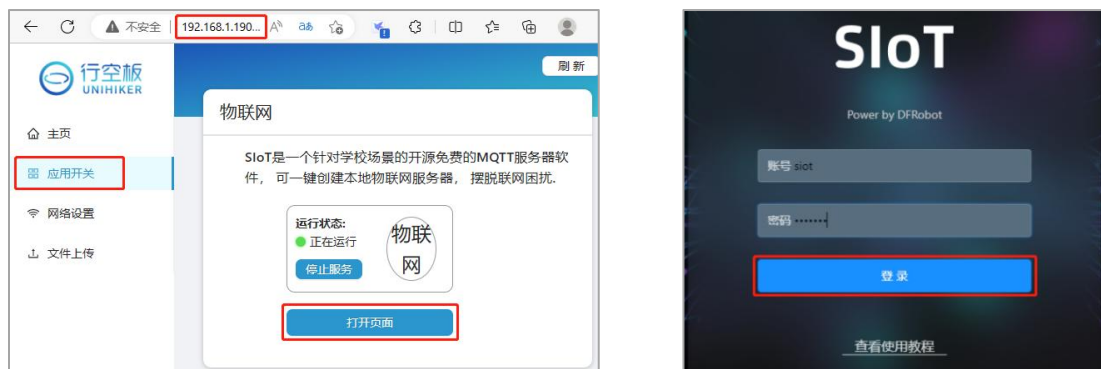
2. 准备工作

- (1) 配置网络：要实现智能终端与智能终端的数据传输，首先要将这两块行空板接入同一网络下，为了可以通过电脑访问 SIoT 服务器，电脑也需要接入同一网络下。

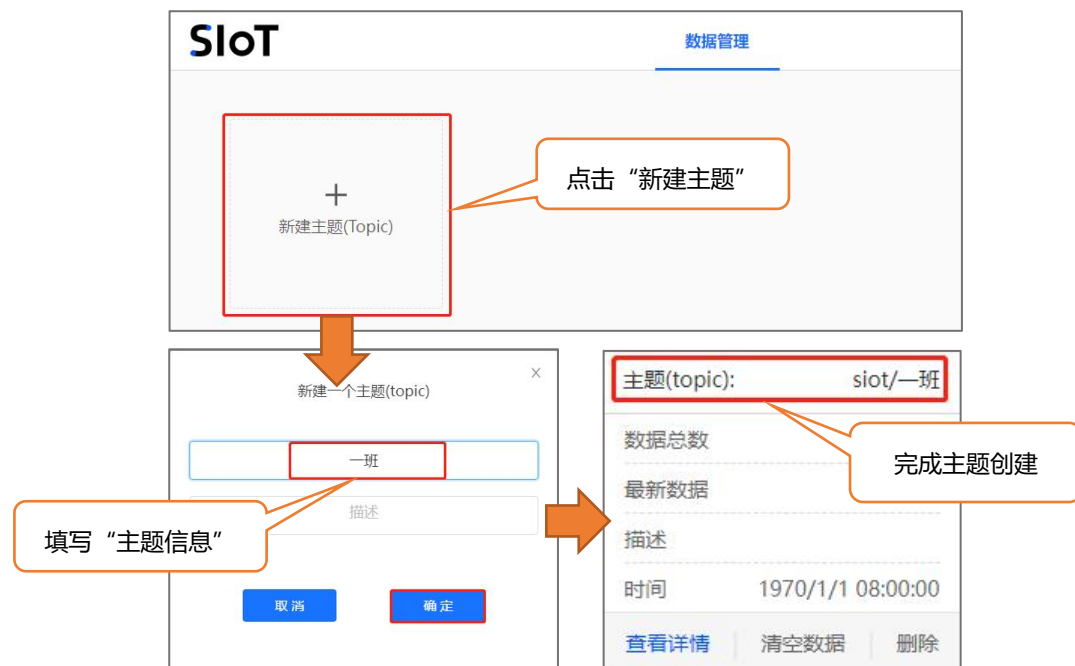
注意：详细的网络配置方法，参考行空板入门套件第 10 课。



- (2) 进入 SIoT2.0 物联网平台：浏览器地址栏输入服务器端 IP 地址：192.168.1.190，进入行空板设置界面，进入应用开关，然后点击“打开页面”，进入 SIoT2.0 服务平台登录页面。输入对应的账号和密码，即可完成登录。**账号：**siot，**密码：**dfrobot。



- (3) 创建主题：登录成功后，进入 SIoT 数据管理页面。然后点击“新建主题”，在弹出窗口中填写“主题名称”，点击“确定”完成主题的创建，页面会显示对应的主题标签。



- (4) 制作课程信息表：新建 excel 表格，然后根据下列表格中的内容进行课程表的制作，最后将工作簿名保存为“高中课表.xls”。

	A	B	C	D	E	F	G
1	节次	星期	星期一	星期二	星期三	星期四	星期五
2	上午	1	语文	数学	英语	语文	品德
3		2	品德	英语	数学	品德	英语
4		3	数学	品德	体育	数学	语文
5		4	英语	语文	语文	英语	数学
6	下午	5	语文	数学	品德	语文	美术
7		6	劳动	美术	数学	数学	活动

3. 软件准备

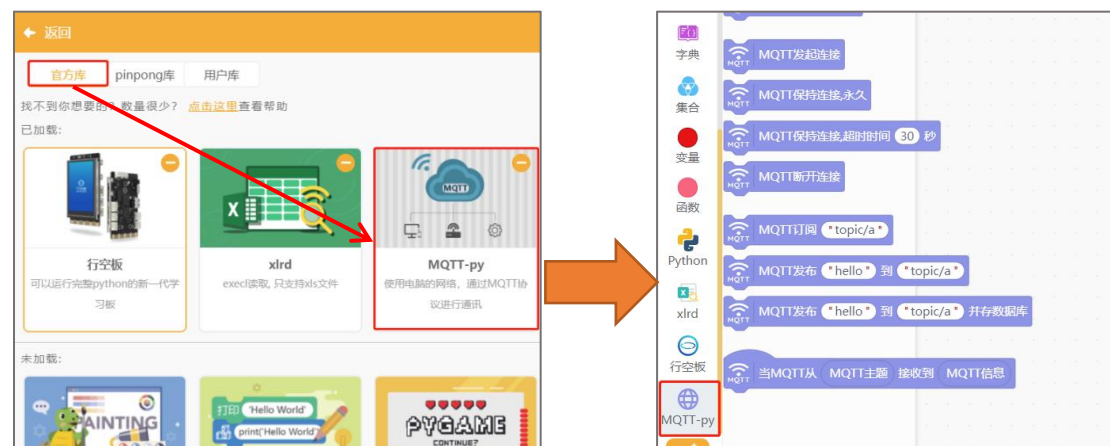
- (1) 打开 Mind+，按照下面图示完成软件准备过程。



(2) 添加“excel 表格读取库”，点击“扩展”，在“官方库”中找到“xlrd”库并点击，完成添加。



(3) 添加“MQTT-py”库，点击“扩展”，在“官方库”中找到“MQTT-py”库并点击，完成添加。



(4) 课程表制作完成后，将“表格素材”文件夹中的“高中课表”，导入到文件系统中。



4. 编写程序

获取工作表中的数据

(1) 要获取工作表中的数据，首先需要使用文件读取“打开 Excel 工作簿（Mind+.xls）为 fp”指令，打开名为“高中课表.xls”的工作簿。



(2) 打开表格后，怎样才能获取表格里的内容呢？使用“获取 fp 的索引为 (0) 的工作表对象 (table)”指令，获取表格中的第一个工作表。

注意：关于工作表和工作簿的区别，见知识园地。



(3) 现在我们要获取工作表中星期一这一列的课程信息，该怎么做呢？通过工作表来分析一下。列号 A、B、C.....对应的就是第 0 列、第 1 列、第 2 列.....；行号 1、2、3.....对应的就是第 0 行、第 1 行、第 2 行。因此，星期一这一列的所有数据，对应的就是第 2 列，7 行数据。

	A	B	C	D	E	F	G
1	节次	课程	星期一	星期二	星期三	星期四	星期五
2	上午	1	语文	第 2 列, 前 7 行内容。		语文	品德
3		2	品德			品德	英语
4		3	数学	品德	体育	数学	语文
5		4	英语	语文	语文	英语	数学
6	下午	5	语文	数学	品德	语文	美术
7		6	劳动	美术	数学	数学	活动

- (4) 接下来, 使用列的操作 “获取 (table) 第 (2) 列的从第 (0) 行到第 (7) 行的数据值” 指令, 获取第 2 列, 从第 0 行到第 7 行的数据。并使用 “打印” 指令, 将这一列的所有信息, 以列表的形式打印出来。

第几列

起始行

终止行

获取 table 第 2 列的从第 0 行到第 7 行的数据值

打印 获取 table 第 2 列的从第 0 行到第 7 行的数据值

终端 (行空板)

```

已终止
root@unihiker:~/mindplus/cache/移动终端.mp# 项目移动终端.mp正在
上传中, 请稍后...
项目上传成功, 开始运行
root@unihiker:~/mindplus/cache/移动终端.mp# access control disa
bled, clients can connect from any host
root@unihiker:~/mindplus/cache/移动终端.mp# cd "/root/mindplus/
cache/移动终端.mp"
root@unihiker:~/mindplus/cache/移动终端.mp# python /root/mindpl
us/cache/移动终端.mp/.cache-file.py
连接结果: 连接成功
['星期一', '语文', '品德', '数学', '英语', '语文', '劳动']
  
```

- (5) 由于, 要将课表信息上传到 SIoT 平台。因此, 新建 “变量课程信息”, 用来存储获取的课程数据。

Python主程序开始

打开Excel工作簿 "高中课表.xls" 为 fp

获取 fp 的 索引 为 "0" 的工作表对象 table

循环执行

将变量 课表信息 赋值为 获取 table 第 2 行的从第 0 列到第 2 列的所有信息

将数据上传到 SIoT 平台

- (1) 新建 “变量时间更新”, 使用 “获取系统时间 (时)” 指令, 获取当前系统时间的小时。第一节的时间是早上 8 点, 那一般 7 点半左右, 同学们就陆续到达教室了。因此, 课程表信息在早上 7 点, 就应该发布到各班显示。



- (2) 通过列确定获取星期几的课表信息，新建“变量星期几”，然后使用“获取系统时间（本周第几天）”指令，来确定要获取工作表中第几列的数据。

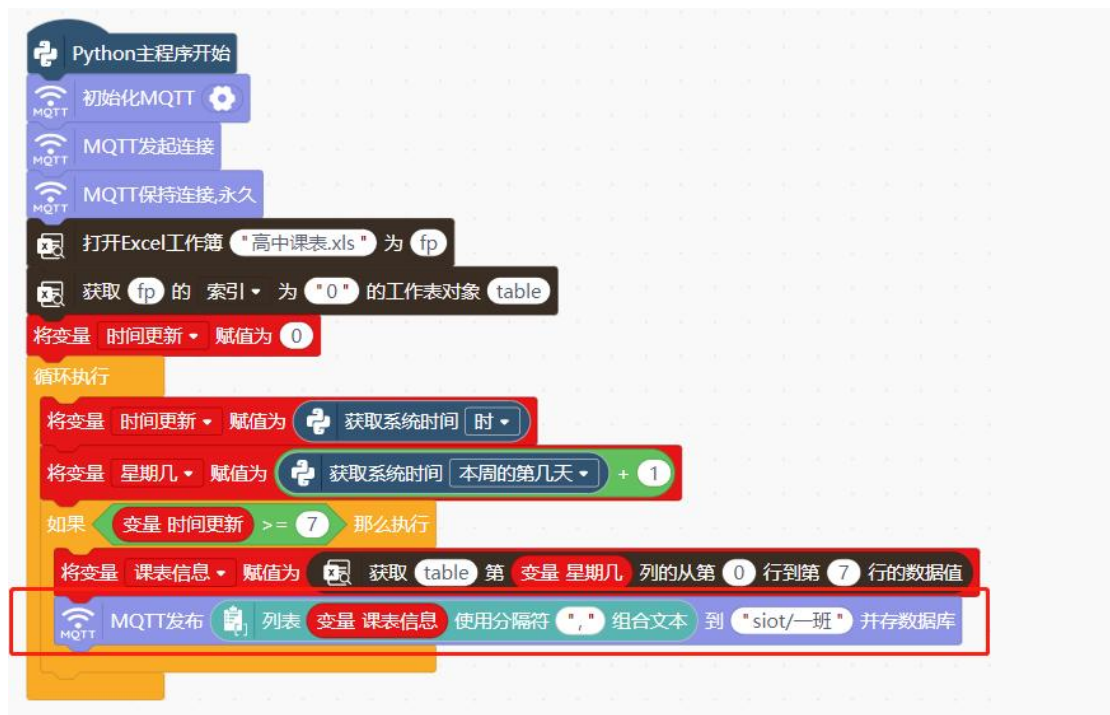


- (3) 课程表信息获取完成后，在“Python 主程序开始”指令下，使用完成“MQTT 的初始化——发起连接——保持连接”三条指令，完成 MQTT 连接三个步骤。



- (4) 这里，使用“列表（）使用分隔符（“，”）组合文本”指令，将“列表课程信息”中的列表数据，

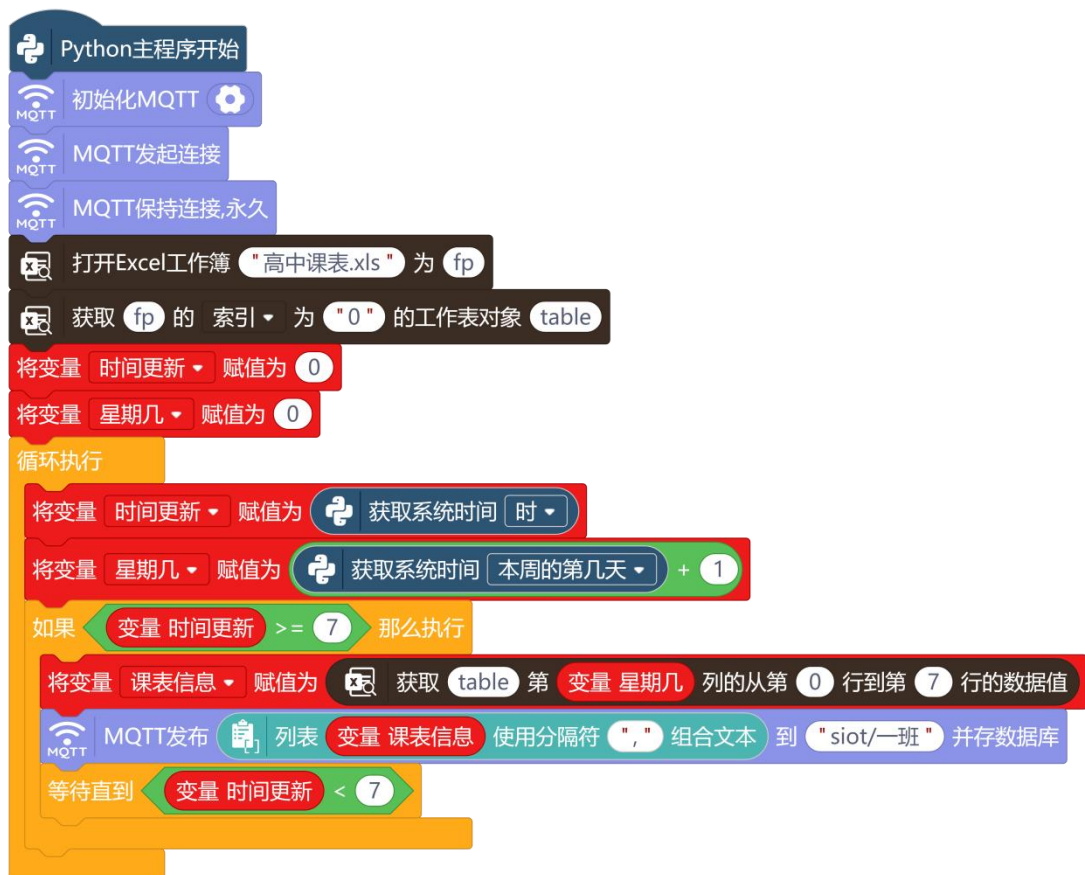
处理成文本数据后。使用“MQTT发布()到()并存数据库”指令，将课表信息发布到主题“siot/一班”中。



- (5) 现在只要当“变量时间更新” ≥ 7 时，就会一直向主题“siot/一班”发送数据。如何才能实现，只发送一次课表信息到主题“siot/一班”中呢？使用“等待直到”与“逻辑运算符 $<$ ”指令，实现每天只发送一次课表信息到主题“siot/一班”。



5. 参考程序



6. 程序运行

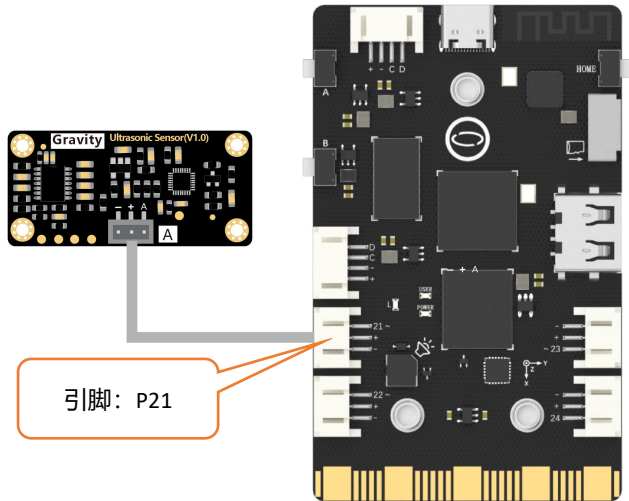
远程连接 192.168.1.190，然后点击右上角的运行按钮，程序运行成功后。移动终端通过识别电脑系统时间以及对应的星期，从工作表中获取对应的课表信息，上传到 SiOT 平台的“siot/一班”主题中。点击主题“siot/一班”中的“查看详情”，就可以查看具体的课程信息了。



任务二：智能终端统计实到人数并显示获取的课表信息

1. 硬件连接

使用连接线将模拟量超声波传感器，连接到行空板的 P21 引脚，如下图。硬件连接成功后，使用 USB 线将行空板连接到计算机。



2. 软件准备

- (1) 打开 Mind+, 按照任务一完成软件的准备过程。
- (2) 添加 MQTT-py 库，点击“扩展”，在“官方库”中找到“MQTT-py”并点击，完成添加。
- (3) 添加超声波库，点击“扩展”，在“pinpog 库”中找到“模拟量超声波测距传感器”并点击，完成添加。



3. 编写程序

在开始编写程序之前，先来分析一下，这个任务需要完成哪些功能。首先，需要设计 UI 界面显示；然后将接收的课表信息，显示在对应的位置；最后，在行空板的最下方位置显示应到人数和实到人数。接下来，就从设计 UI 界面、显示课表信息、显示实到人数这三个功能来完成。

设计 UI 界面



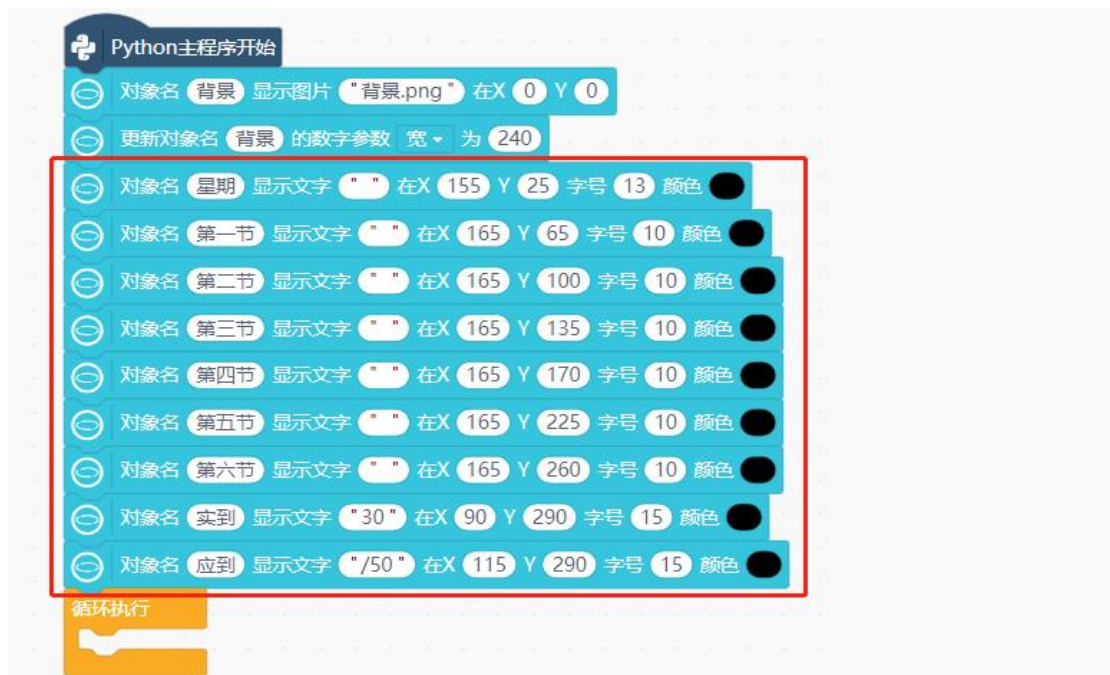
- (1) 为了降低编程难度，将 UI 界面中不用改变的文字制作成了一张背景图。现在，将“图片素材”文件夹中的“背景.png”导入到文件系统中。



- (2) 使用“对象名 () 显示图片 () 在 x (0) y (0)”指令，将背景.png 显示在行空板上。然后，使用“更新对象名 () 的数字参数宽为 ()”指令，设置背景图片的宽为 240。



- (3) 使用“对象名 () 显示文字 () 在 x () y () 字号 () 颜色 ()”指令，设置星期、第一节课、第二节课.....的信息，显示在对应的位置。



获取课表信息

- (1) 要获取课表信息，首先在“Python 主程序开始”指令下，使用完成“MQTT的初始化——发起连接——保持连接”三条指令，完成 MQTT 连接三个步骤。然后使用“MQTT 订阅 ()”指令，订阅主题“siot/一班”。



- (2) 新建“变量显示课表”、“变量课表信息”，并在“Python 主程序开始”指令下，初始化“变量显示课表”的值为 0，初始化“变量课表信息”为空列表。然后，在“当 MQTT 从 MQTT 主题接收到 MQTT 信息”指令下，将“变量显示课表”、“变量课表信息”定义为“全局变量”。



- (3) 新建“变量接收信息”当智能终端接收到有 MQTT 信息时，就将“MQTT 信息”赋值给“变量接收信息”。



- (4) 由于，接收到的 MQTT 信息为文本型。为了方便后面显示每节课的课表信息，使用“文本 () 使用分隔符 (“,” ”) 制作列表”指令，将“变量接收信息”制作成列表赋值给“变量课表信息”，并将“变量显示课表”赋值为 1。



- (5) 在“循环执行”指令中，判断变量显示课表的值是否为 1，当条件为真时，说明接收到教务处发送的课表信息。使用“列表 () 索引 () 的值”指令，将“列表课表信息”中的数据索引出来。并使用“更新对象名 () 的文本内容参数为 ()”指令，将显示内容，更新为课表信息。最后，将“变量显示课表”赋值为 0。



统计实到人数

- (1) 在“Python 主程序开始”指令下，使用“模拟量超声波 (urm091) 初始化 引脚”指令，初始化引脚为“行空板用脚 (P21~A)”。然后，新建“变量距离”、“变量人数”并初始化为 0。

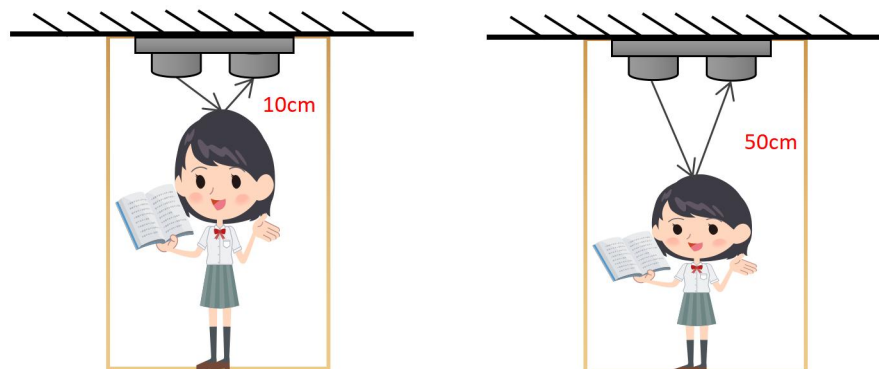
注意：关于超声波传感器的使用方法及原理，见知识园地。



- (2) 在“循环执行”指令中，使用“(urm091) 读取模拟量超声波 (cm)”指令，获取超声波传感器读取到的距离，并赋值给“变量距离”。



- (3) 超声波传感器安装在门框上方，当检测到的距离大于 10 且小于 50cm 的时候（假设学生的身高最高为 1.9m，最低为 1.5m），说明有学生进入教室。



- (4) 新建“变量实到人数”并在“Python 主程序开始”下，初始化值为 0。使用“条件判断”指令，判断变量距离是否大于 10 且小于 50，当条件为真时，接着使用“条件判断”指令，判断变量实到人数是否小于 50，当条件为真时，就让“变量实到人数”+1。



- (5) 最后，使用“更新对象名（）的文本内容参数为（）”指令，更新对象名应到的文本内容参数为“变量实到人数”。



4. 参考程序

Python主程序开始

模拟量超声波 urm091 初始化 引脚 行空板引脚 P21(~A)

初始化MQTT

MQTT发起连接

MQTT保持连接,永久

MQTT订阅 "siot/一班"

将变量 距离 赋值为 0

将变量 实到人数 赋值为 0

将变量 显示课表 赋值为 0

将变量 课表信息 赋值为 初始化列表 []

对象名 背景 显示图片 "背景.png" 在X 0 Y 0

更新对象名 背景 的数字参数 宽 为 240

对象名 星期 显示文字 "" 在X 155 Y 25 字号 13 颜色

对象名 第一节 显示文字 "" 在X 165 Y 65 字号 10 颜色

对象名 第二节 显示文字 "" 在X 165 Y 100 字号 10 颜色

对象名 第三节 显示文字 "" 在X 165 Y 135 字号 10 颜色

对象名 第四节 显示文字 "" 在X 165 Y 170 字号 10 颜色

对象名 第五节 显示文字 "" 在X 165 Y 225 字号 10 颜色

对象名 第六节 显示文字 "" 在X 165 Y 260 字号 10 颜色

对象名 实到 显示文字 变量 实到人数 在X 90 Y 290 字号 15 颜色

对象名 应到 显示文字 "/50" 在X 115 Y 290 字号 15 颜色

循环执行

将变量 距离 赋值为 urm091 读取模拟量超声波(cm)

如果 变量 距离 >= 10 与 变量 距离 <= 50 那么执行

如果 变量 实到人数 < 50 那么执行

将变量 实到人数 赋值为 变量 实到人数 + 1

更新对象名 实到 的文本内容参数为 变量 实到人数

等待 0.5 秒

如果 变量 显示课表 = 1 那么执行

更新对象名 星期 的文本内容参数为 列表 变量 课表信息 索引 0 的值

更新对象名 第一节 的文本内容参数为 列表 变量 课表信息 索引 1 的值

更新对象名 第二节 的文本内容参数为 列表 变量 课表信息 索引 2 的值

更新对象名 第三节 的文本内容参数为 列表 变量 课表信息 索引 3 的值

更新对象名 第四节 的文本内容参数为 列表 变量 课表信息 索引 4 的值

更新对象名 第五节 的文本内容参数为 列表 变量 课表信息 索引 5 的值

更新对象名 第六节 的文本内容参数为 列表 变量 课表信息 索引 6 的值

将变量 显示课表 赋值为 0

当MQTT从 MQTT主题 接收到 MQTT信息

全局global 变量 显示课表

全局global 变量 课表信息

将变量 接收信息 赋值为 MQTT信息

将变量 课表信息 赋值为 文本 变量 接收信息 使用分隔符 "," 制作列表

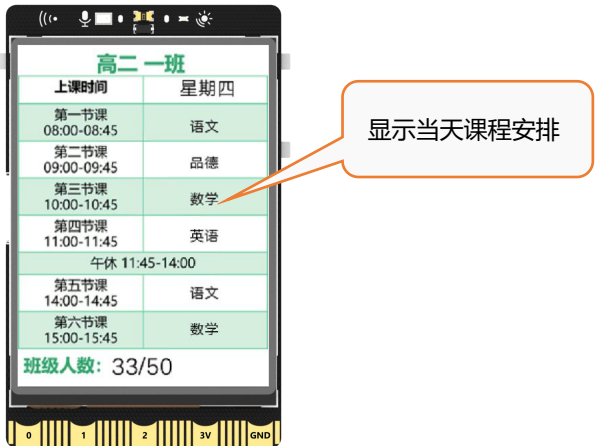
将变量 显示课表 赋值为 1

5. 程序运行

智能终端远程连接 192.168.1.208，然后点击右上角的运行按钮，程序运行成功后。智能终端可以根据模拟量超声波传感器统计班级实到人数。



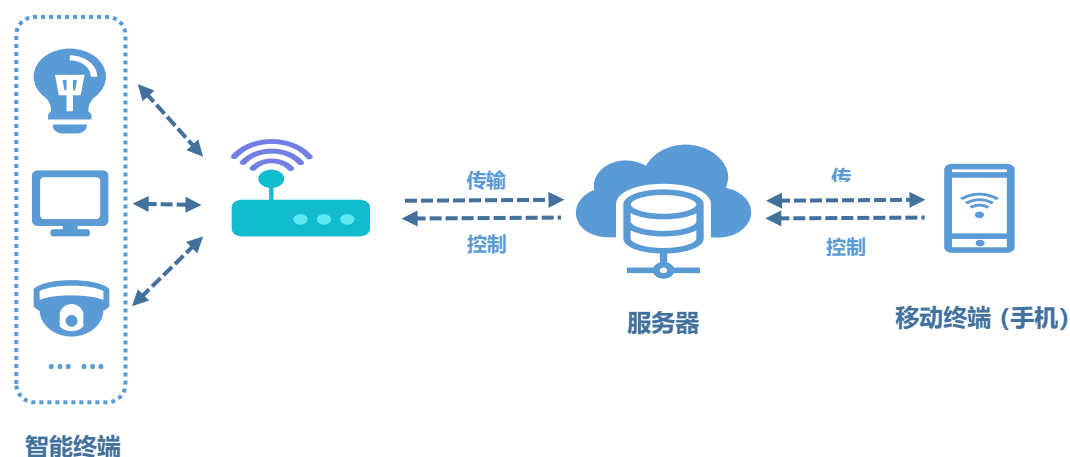
移动终端远程连接 192.168.1.190，然后点击右上角的运行按钮，程序运行成功后。根据系统时间获取工作表中当天对应的课程信息，并上传到 IoT 平台。智能终端接收到课程信息后，将当天的课程安排显示对应位置。



知识园地

1. 了解什么智能终端和移动终端

一个完整的物联网系统包含三个部分，分别是服务器、智能终端、移动终端。以智能家居场景为例，我们可以通过手机控制和查看灯、风扇、摄像头这些设备的状态，手机就是物联网系统中的**移动终端**；而那些可以直接连接网络的灯、摄像头等智能家电就是物联网系统中的**智能终端**；另外，在物联网系统中，移动终端和智能终端不能直接传输数据，它们之间还需要一个枢纽，用来存储和收发数据，这就是**服务器**。



2. 了解常见的通讯方式

课程中最常用到的通讯方式有：有线通讯、无线通讯、远程通讯。下面，就来了解一下这三种常用的通讯方式吧！

有线通讯是使用电缆或光缆连接设备之间的通讯当时，例如以太网、串口通讯、USB 等。使用 USB 线将行空板连接到电脑然后运行程序，这种方式就是有线通讯。



无线通讯是使用无线电波或红外线等无线信号进行通讯，例如无线局域网（WiFi）、蓝牙、红外线等。使用行空板向 SIoT 平台发送数据，这种方式就是无线通讯。



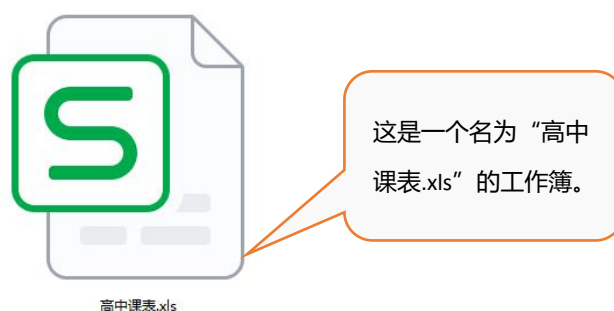
主题(topic):	siot/一班
数据总数	1
最新数据	星期三,英语,数学...
描述	
时间	2023/4/26 15:22:13
查看详情 清空数据 删除	

远程通讯是利用网络的形式进行远程通讯，例如互联网、局域网、广域网等。在第 10 课中，通过移动终端的行空板控制智能终端上的设备关闭或开启就属于远程通信。当然，这节课中智能终端接收移动终端读取的课表信息也属于远程通讯。



3. 了解工作簿与工作表的区别

什么是工作簿？工作簿是一个文件，它可以包含多个工作表。每个工作簿都有一个对应的文件名，并以文件的形式存储在计算机硬盘上。

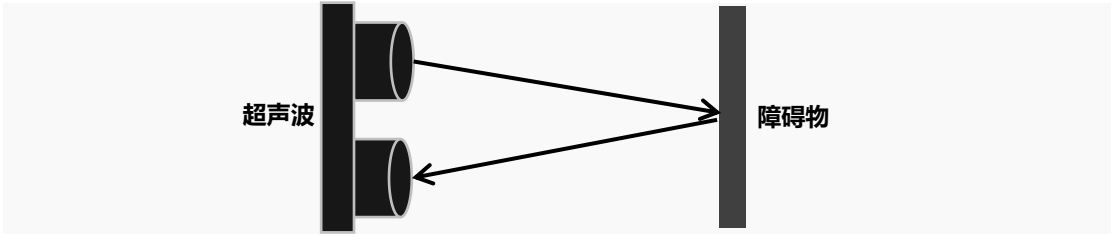


什么是工作表？工作表是工作簿中一个单独的页面，由行和列组成，可以存储和展示数据。每个工作表也有自己的名称，便于区分不同的数据表。在一个工作簿中，可以有多个工作表，每个工作表都可以存储不同的数据。例如，在“高中课表.xls”工作簿中，有三个名为 Sheet1、Sheet2、Sheet3 的工作表。

		星期一	星期二	星期三	星期四	星期五
节次	课程	语文	数学	英语	语文	品德
上午	1	语文	数学	英语	语文	品德
	2	品德	英语	数学	品德	英语
	3	数学	品德	体育	数学	语文
	4	英语	语文	语文	英语	数学
下午	5	语文				美术
	6	劳动				活动

4. 了解超声波传感器的工作原理

模拟量超声波测距传感器是一款开放式双探头的超声波测距模块，输出模拟电压值。这个模块是如何实现测量距离的呢？超声波模块有两个探头，一个探头用来发射信号，另一个探头用来接收遇到障碍物返回的电压信号。



将返回后的电压信号处理后，就可以获取超声波检测到与障碍物的距离。使用“(urm091) 读取模拟量超声波 (cm)”指令，获取距离，其实就是在获取的处理后的电压信号。

5. 指令学习

Block	功能说明
	该指令用于打开 Excel 名为.xls 的工作簿，并将转换后的工作簿对象赋值给变量 fp。
	该指令用于通过不同的值来索引工作簿中的工作表，并将工作表转换后赋值给变量 table。
	该指令用于提取工作表中指定行列的数据。
	该指令用于获取系统时间，可以获取年、月、日、时、分、秒、本周的第几天、本年的第几天。
	该指令用于将列表中的数据，通过分隔符的方式将数据组合成文本。
	该指令用于将文本数据，通过分隔符的方式制作成列表。
	该指令用于初始化模拟量超声波传感器连接在行空板上的引脚。
	该指令用于读取模拟量超声波传感器检测到的距离。

挑战自我

这节课中，移动终端获取的是第一个工作表中的课程信息，在提供的表格素材“高中课程.xls”工作簿中有三个工作表，并且每个工作表中的课程内容是不一样的。接下来，大家使用上面学习过的方法，修改任务一中的程序，实现获取其余两个工作表中的课程内容！

提示：修改该指令的索引序号，即可获取不同工作表中的内容。



第 13 课 IoT 游戏排行服务器

游戏排行服务器是一种在线游戏服务，主要用于收集玩家在游戏的成绩和排名，然后将排名后的成绩存储到服务器上，供玩家和其他用户查看。

这节课，主要是为第 6 课障碍小游戏，制作一款可以根据游戏时长排名的 IoT 游戏排行服务器。实现游戏结束后统计游戏时长，然后上传玩家的游戏 ID 和游戏时长到 IoT 游戏排行服务器上，服务器对时长排名后，显示在可视化界面上。



任务目标

智能终端在游戏结束后，上传自己的游戏 ID 与游戏积分，然后服务器端接收游戏 ID 后游戏积分后，对积分进行降序排序，最后利用 Mind+的可视化面板，将排序后的前三名游戏积分显示在对应名次上。



知识点

- 1. 了解多设备的组网技术
- 2. 掌握字典的使用方法
- 3. 学习列表中的排序

材料清单

硬件清单：

	
行空板 x2	按钮模块 x1
	
白色硅胶绞线 x1	Type-C USB 线 x2

软件使用：Mind+编程软件（1.8.0 及以上） x1

下载地址：<https://mindplus.cc/>

动手实践

IoT 游戏排行服务器，主要是利用 Mind+可视化界面，制作一个可显示游戏积分排名情况的界面。首先，要在第 6 课程序的基础上，增加统计积分功能，然后玩家游戏结束后，将积分上传到服务器端。服务器端对上传的积分进行排序并显示在可视化界面上。接下来，就从下面三个任务来完成吧！

任务一：智能终端向 SIoT 服务器端发送游戏积分

在第 6 课任务三的程序上，增加统计游戏时长功能。当游戏结束后，将对应的时长积分发送给 SIoT 服务器平台。

任务二：搭建 SIoT 游戏排行服务器

接收玩家上传的游戏 ID 和游戏积分，然后对游戏积分进行降序排序。

任务三：制作游戏排行可视化界面

利用 Mind+的可视化面板，制作游戏排行界面，然后将排序后的前三名游戏积分显示在对应名次上。

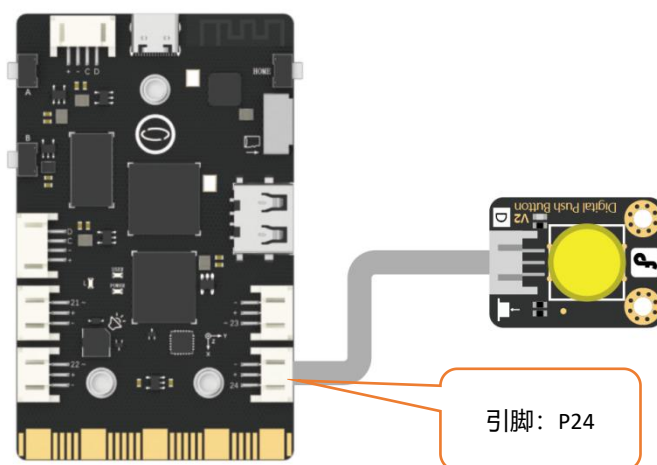
任务一：智能终端向 SIoT 服务器端发送游戏积分

要实现智能终端（玩家端）与服务器端（游戏积分排行服务器）的数据传输，需要至少两块行空板来搭建物联网系统。当游戏结束后，智能终端向 SIoT 服务器端发送游戏积分。

注意：我们可以使用老师的行空板作为服务器端。

1. 硬件连接

使用连接线将按钮模块连接到行空板的 P24 引脚，如下图。硬件连接成功后，使用 USB 线将行空板连接到计算机。



2. 准备工作

- (1) 配置网络：要实现移动与智能终端的数据传输，首先要将这两块行空板接入同一网络下，为了可以通过电脑访问 SIoT 服务器，电脑也需要接入同一网络下。

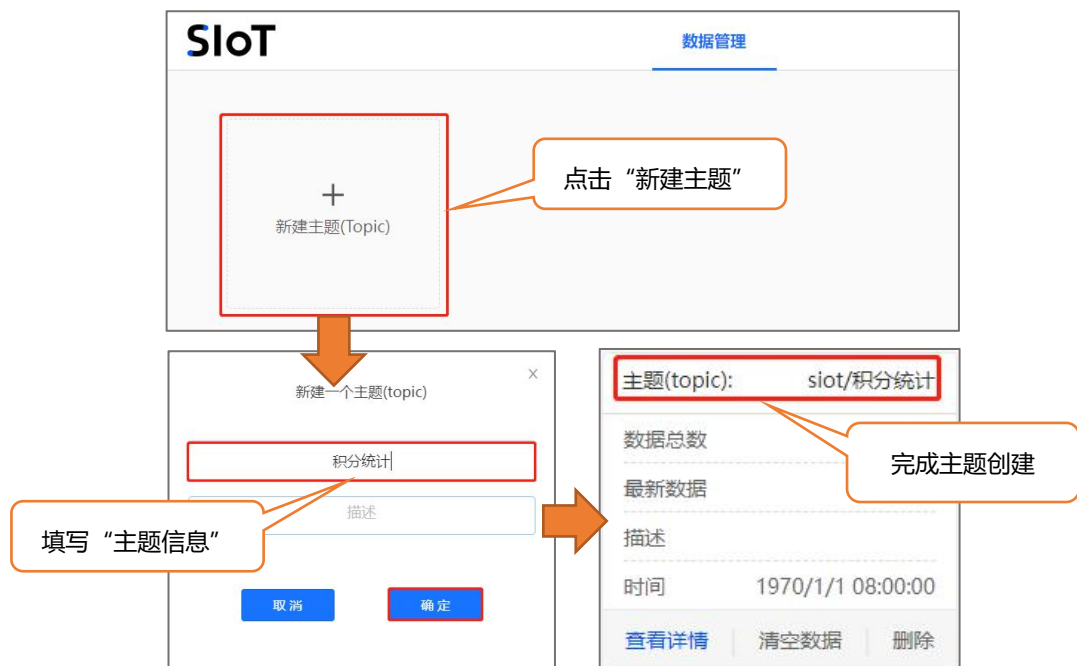
注意：详细的网络配置方法，参考行空板入门套件第 10 课。



- (2) 进入 SIoT2.0 物联网平台：浏览器地址栏输入服务器端 IP 地址：192.168.1.190，进入行空板设置界面，进入应用开关，然后点击“打开页面”，进入 SIoT2.0 服务平台登录页面。输入对应的账号和密码，即可完成登录。**账号：**siot，**密码：**dfrobot。



- (3) 创建主题：登录成功后，进入 SIoT 数据管理页面。然后点击“新建主题”，在弹出窗口中填写“主题名称”，点击“确定”完成主题的创建，页面会显示对应的主题标签。



3. 软件准备

- (1) 打开 Mind+, 按照下面图示完成软件准备过程。



(2) 点击添加 MQTT-py 库，点击“扩展”，在“官方库”中找到“MQTT-py”并点击，完成添加。

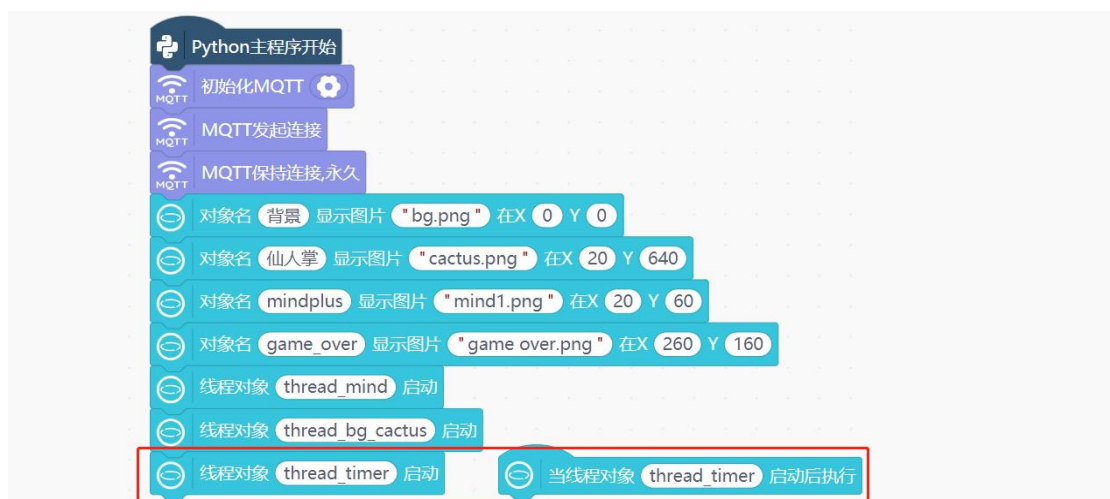


4. 编写程序

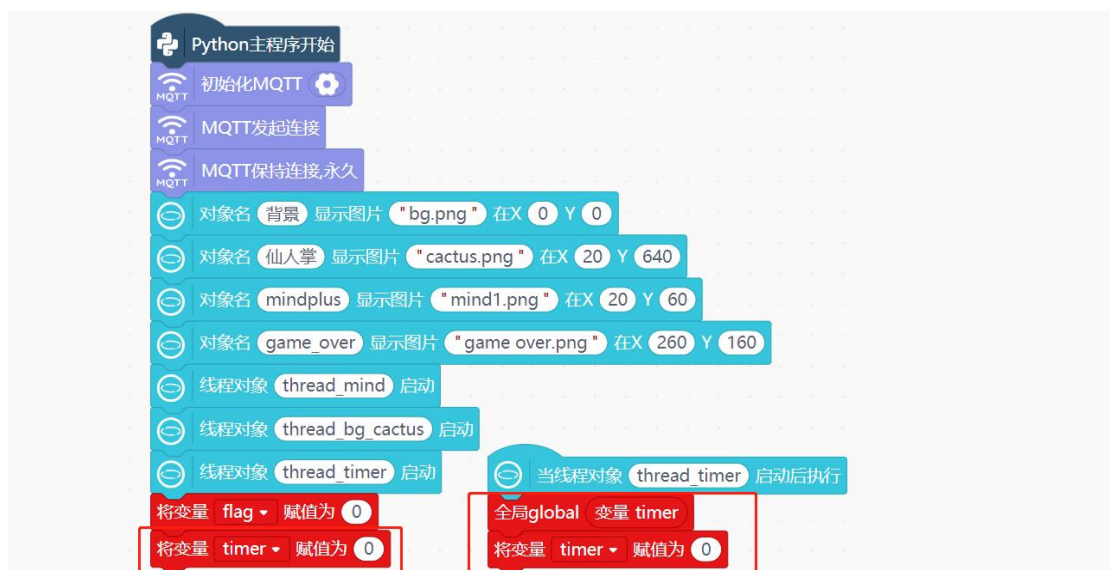
(1) 这个任务主要是在第 6 课任务三的程序基础上，添加游戏计时的功能。首先在“Python 主程序开始”指令下，使用完成“MQTT 的初始化——发起连接——保持连接”三条指令，完成 MQTT 连接三个步骤。



- (2) 使用“当线程对象 (thread) 启动后执行”指令, 新建一个 thread_timer 线程, 并在程序“线程对象 thread_bg_cactus 启动”指令下, 使用“线程启动”指令, 启动 thread_timer 线程。



- (3) 新建“变量 timer”, 并在 Python 主程序开始”指令下, 初始化变量 timer 的值为 0, 然后在“当线程对象 (thread_timer) 启动后执行”指令下, 定义变量 timer 为全局变量, 再次初始化变量 timer 的值为 0。



- (4) 如何才能实现游戏计时呢？在“当线程对象 (thread_timer) 启动后执行”指令下，添加“循环执行”指令。每隔一秒，就让变量 timer+1，实现计时功能。



- (5) 在游戏结束后，使用“线程对象 (thread_timer) 停止”指令，终止 thread_timer 线程。并使用“MQTT 发布 () 到 () 并保存数据库”指令、“合并字符串”指令，将玩家 ID 和游戏积分进行合并后，一起发布在 IoT 服务器端。

注意：为了便于后面对游戏 ID 与游戏积分进行区分，游戏 ID 只能为四个字符。



- (6) 游戏结束后，只需要向 IoT 游戏排行服务器端发送一次游戏积分数据，因此，增加一个标志变量“变量 flag_2”，并初始化变量值为 1，当变量 flag_2 的值为 1 时，再来判断是否遇到障碍物，当游戏积分数据发送成功后，修改变量 flag_2 的值为 0。



5. 参考程序

```

Python主程序开始
初始化MQTT
MQTT发起连接
MQTT保持连接,永久

对象名 背景 显示图片 "bg.png" 在X 0 Y 0
对象名 仙人掌 显示图片 "cactus.png" 在X 20 Y 640
对象名 mindplus 显示图片 "mind1.png" 在X 20 Y 60
对象名 game_over 显示图片 "game over.png" 在X 260 Y 160

线程对象 thread_mind 启动
线程对象 thread_bg_cactus 启动
线程对象 thread_timer 启动

将变量 flag 赋值为 0
将变量 timer 赋值为 0
将变量 flag_2 赋值为 1

循环执行
如果 变量 flag = 0 那么执行
    如果 读取数字引脚 P24 = 1 那么执行
        将变量 flag 赋值为 1
        等待 0.01 秒
    如果 变量 flag = 1 那么执行
        将变量 flag 赋值为 0
        更新对象名 mindplus 的数字参数 x 为 90
        等待 0.4 秒
        更新对象名 mindplus 的数字参数 x 为 20
    如果 变量 flag_2 = 1 那么执行
        如果 变量 cactus_y = 60 那么执行
            更新对象名 game_over 的数字参数 x 为 200
            更新对象名 game_over 的基准点为 中心
            线程对象 thread_bg_cactus 停止
            线程对象 thread_mind 停止
            线程对象 thread_timer 停止
        MQTT发布 合并 "时时木" 变量 timer 到 "siot/积分统计" 并存数据库
        将变量 flag_2 赋值为 0
        将变量 flag 赋值为 2

当线程对象 thread_timer 启动后执行
全局global 变量 timer
将变量 timer 赋值为 0
循环执行
    等待 1 秒
    将变量 timer 赋值为 变量 timer + 1

当线程对象 thread_mind 启动后执行
循环执行
    更新图片对象 mindplus 图片来源为 "mind2.png"
    等待 0.1 秒
    更新图片对象 mindplus 图片来源为 "mind3.png"
    等待 0.1 秒
    更新图片对象 mindplus 图片来源为 "mind1.png"
    等待 0.1 秒

当线程对象 thread_bg_cactus 启动后执行
全局global 变量 cactus_y
全局global 变量 flag_1
将变量 flag_1 赋值为 0
将变量 cactus_y 赋值为 640
循环执行
    如果 变量 cactus_y >= -100 那么执行
        将变量 cactus_y 赋值为 变量 cactus_y - 10
        更新对象名 仙人掌 的数字参数 y 为 变量 cactus_y
    否则
        将变量 cactus_y 赋值为 320
    如果 变量 flag_1 >= -320 那么执行
        将变量 flag_1 赋值为 变量 flag_1 - 10
        更新对象名 背景 的数字参数 y 为 变量 flag_1
    否则
        将变量 flag_1 赋值为 0
    等待 0.02 秒
    
```

6. 程序运行

远程连接 192.168.1.190, 然后点击右上角的运行按钮, 程序运行成功后。当 mind+角色与仙人掌接近时, 按下按钮, 让 mind+越过仙人掌, mind+继续向前进; 如果 mind+没有越过仙人掌, 屏幕上则显示 "game over" 的信息。游戏结束后, 将游戏 ID 与积分发送到 SIoT 服务器端的主题 "siot/积分统计" 中。



主题(topic):	siot/积分统计
数据总数	1
最新数据	时时木69
描述	
时间	2023/4/17 13:45:24
查看详情 清空数据 删除	

任务二：搭建 SIoT 游戏排行服务器

SIoT 游戏排行服务器的主要工作是接收智能终端发送的积分数据，然后对接收到的所有积分进行排序。最后将排序好的积分，按照名次发送到对应的主题中。

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

- (1) 打开 Mind+, 按照任务一完成软件的准备过程。
- (2) 添加 MQTT-py 库, 点击“扩展”, 在“官方库”中找到“MQTT-py”并点击, 完成添加。

3. 编写程序

获取并存储 MQTT 消息

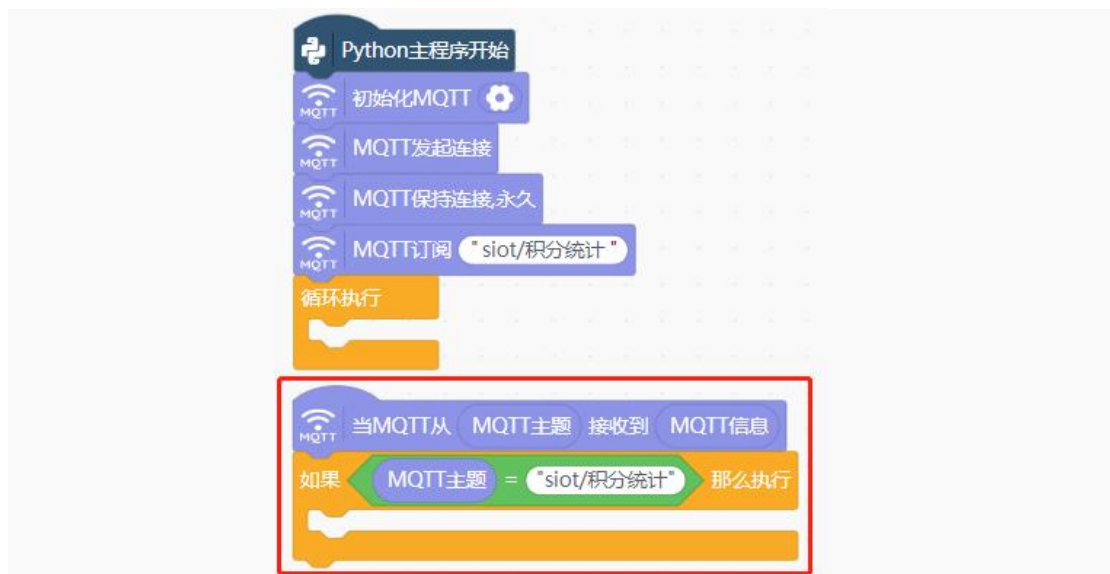
- (1) 在 Python 主程序开始之下, 完成“MQTT 的初始化——发起连接——保持连接”这三个步骤后, 需要将 SIoT 服务器中的 IP 地址修改为: 192.168.1.190。



- (2) 任务一中, 游戏积分上传到 SIoT 服务器主题的“siot/积分统计”后, 游戏排行服务器中要对积分进行统计并排序。需要先订阅主题, 才能接收对应的积分数据。因此, 使用“MQTT 订阅 (topic/a)”指令, 订阅主题“siot/积分统计”。



- (3) 如何才能在 SIoT 游戏排行服务器端接收智能终端发送的游戏 ID 与游戏积分呢？使用“当 MQTT 从 (MQTT 主题) 接收到 (MQTT 信息)”指令，接收数据。然后，使用“条件判断”指令，判断 MQTT 主题是否为“siot/积分统计”。



- (4) 新建一个名为积分统计的字典，并在“Python 主程序开始”指令下，使用“初始化字典{ }”指令，初始化字典内容为空。

注意：关于“字典”的相关介绍及使用方法，见知识园地。



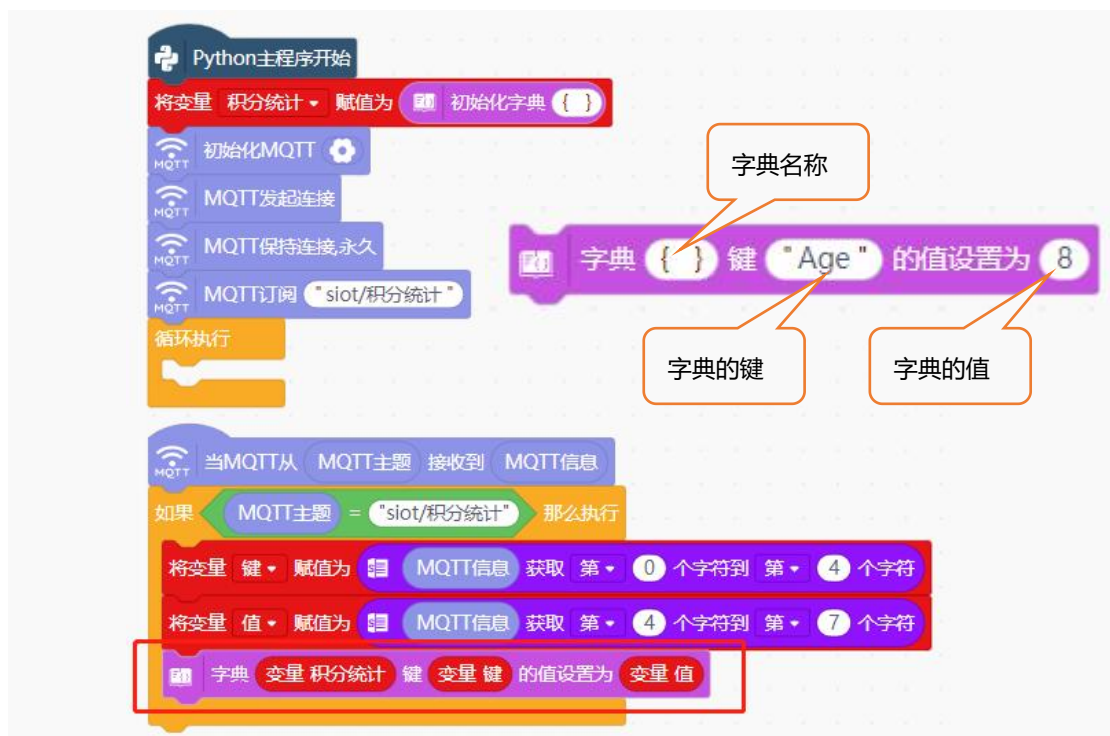
- (5) 向字典存储消息，是按照字典的键和值来存储。如何才能判断 MQTT 消息中的哪部分内容是键，哪部分内容是值呢？接收到的 MQTT 消息是由游戏 ID+游戏积分的形式组成，MQTT 消息前四个字符为游戏 ID，作为字典的键；MQTT 消息后四个字符为游戏积分，作为字典的值。



- (6) 因此，新建“变量键”、“变量值”，使用“获取第 (0) 个字符到第 (4) 个字符”指令，将“MQTT 消息”中的游戏 ID 获取出来，并赋值给变量键；然后使用“获取第 (4) 个字符到第 (7) 个字符”指令，将“MQTT 消息”中的游戏积分获取出来，并赋值给变量值。



- (7) 最后，使用“字典{ }键 () 的值设置为 ()”指令，将赋值后的变量键和变量值，存储在字典积分统计中。

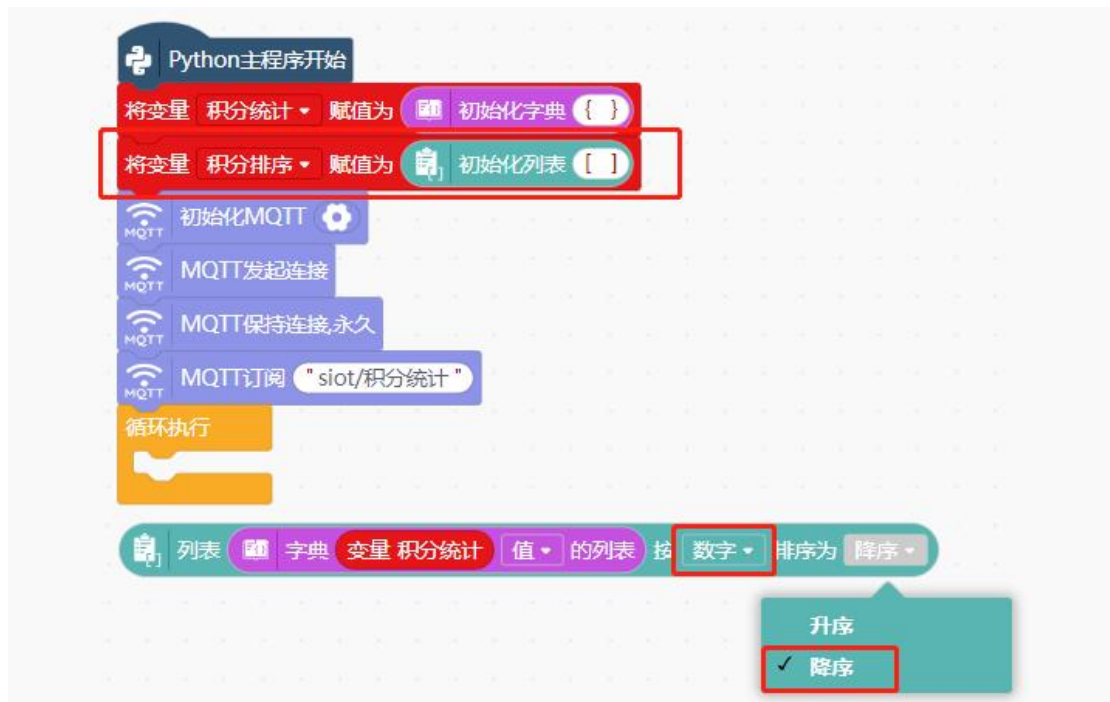


积分排序

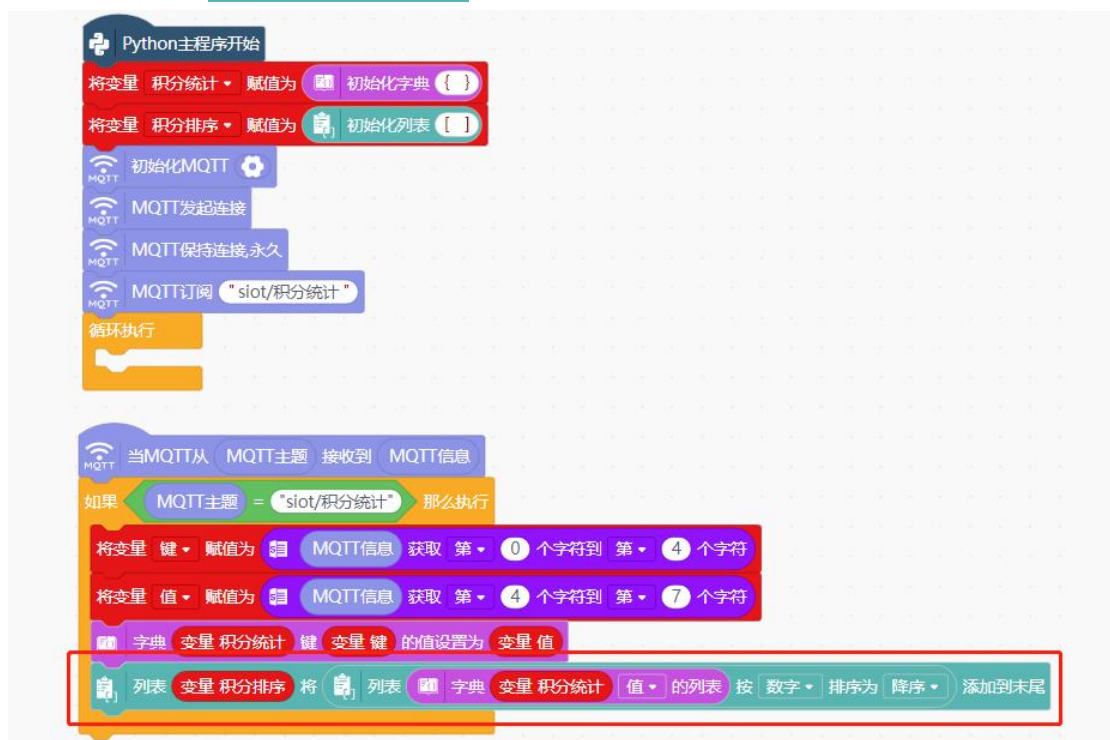
- (1) 积分是字典中的值，如何才能对字典中的值进行排序呢？使用“字典 (键/值) 的列表”指令，将字典的值提取出来。



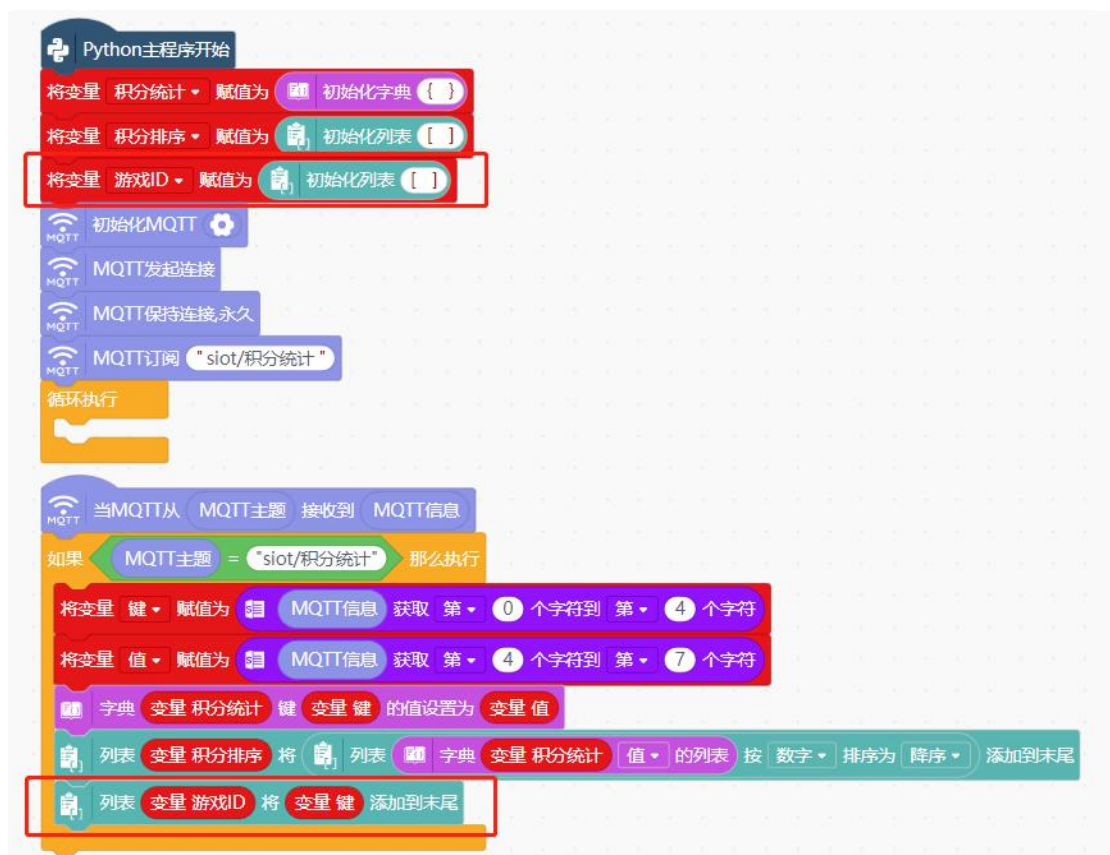
- (2) 新建“列表积分列表”，并在“Python 主程序开始”下，使用“初始化列表[]”指令，初始化列表值为空，使用“列表[]按 (数字) 排序为升序”指令，对字典积分统计值的列表按照数字进行降序排序。



(3) 然后，使用“列表[]将 () 添加到末尾”指令，将排序后的积分添加在列表积分排序中。



(4) 新建“列表游戏 ID”，并在“Python 主程序开始”下，使用“初始化列表[]”指令，初始化列表值为空。然后，将 MQTT 消息中的游戏名称（变量键），使用“列表[]将 () 添加到末尾”指令，添加到列表游戏 ID 中。

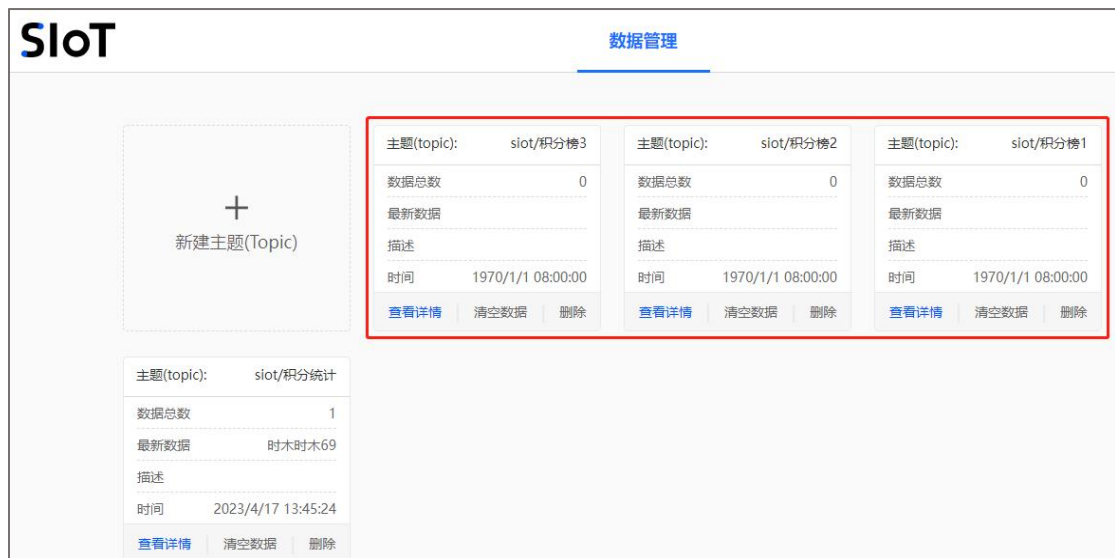


- (5) 然后，使用“列表[]查找 () 的索引”指令，依次将列表积分排序中的最后一个积分数据索引出来，存放在列表排序后积分中。



上传前三名积分到 IoT 游戏排行服务器

- (1) 首先，在 IoT 游戏排行服务器上，分别新建三个主题备用“slot/积分第 1”、“slot/积分第 2”、“slot/积分第 3”。



- (2) 当 SIoT 游戏排行服务器的 “slot/积分统计” 主题中，只有 1 条游戏 ID 与游戏积分数据时，那这条数据，就是积分排行榜的第 1 名。因此，通过 “条件判断” 语句，判断列表排序后积分的长度是否为 1？条件为真时，使用 “MQTT 发布 () 到 () 并保存数据库” 指令，将变量键与变量值合并后，发送到主题 “slot/积分榜第 1”。



- (3) 当 SIoT 游戏排行服务器的 “slot/积分统计” 主题中，有 2 条游戏 ID 与游戏积分数据时，需要将排序后的积分，按照游戏 ID: 游戏积分的方式，依次发送到主题 “slot/积分榜第 1”、“slot/积分榜第 2” 中。如果只发送积分，直接使用 “for 循环” 指令，将列表排序后积分中的数据，挨个提取出来发送即可。



- (4) 现在要按照游戏 ID：游戏积分的方式，发送数据，怎样才能知道排序后的积分对应的游戏 ID 呢？例如，列表排序后积分存放的是{69, 80}，字典积分统计中存放的是[时木时木 69, 是小麦呀 80]，如何才能判断积分第一名：80，对应的游戏 ID 为：是小麦呀？

列表 排序后积分	字典 积分统计 (键)	字典 积分统计 (值)	列表 游戏 ID
80	时木时木	69	时木时木
69	是小麦呀	80	是小麦呀

- (5) 分别新建“变量列表积分”、“变量字典积分”，然后使用“for 循环”指令，将列表排序后积分与字典积分统计中的值提取出来赋值到对应的变量中。



- (6) 使用“条件判断”指令，判断列表积分的值是否等于字典积分中的值。如果条件为真，使用“合并字符串”指令，将索引出对应的游戏 ID 与列表积分进行合并，赋值给“变量游戏 ID 与积分”。

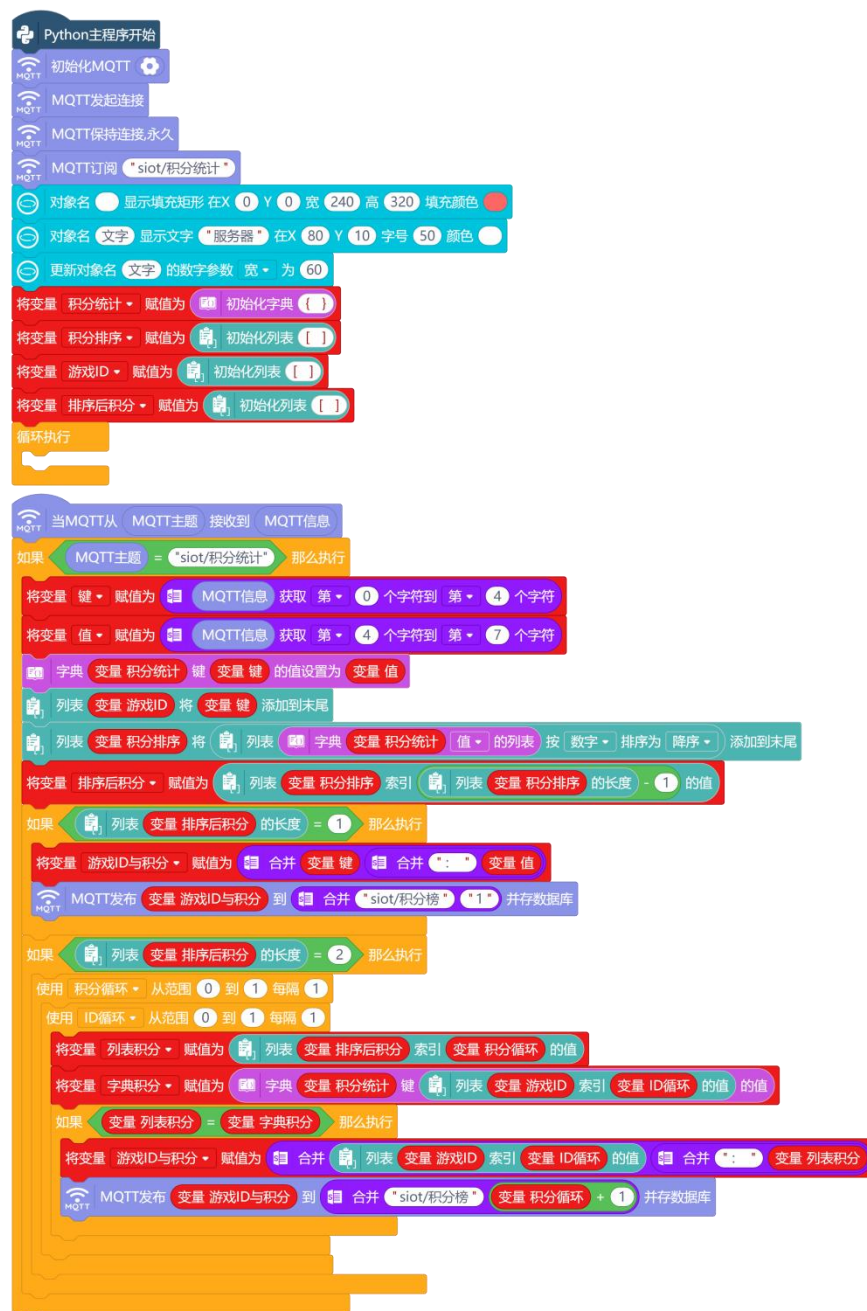


- (7) 使用“MQTT 发布 () 到 () 并保存数据库”指令，将“变量游戏 ID 与积分”，分别发送到主题“siot/积分第 1”、“siot/积分榜 2”中。



(8) 最后, 使用“填充矩形”、“显示文字”、“更新数字参数宽”指令, 在行空板上显示文字“服务器”。

4. 参考程序

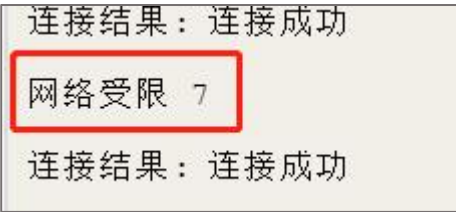


5. 程序运行

先将 SIoT 游戏排行服务器端，远程连接 192.168.1.190，点击右上角的运行按钮。再将智能终端远程连接在 192.168.1.208，点击右上角的运行按钮，程序运行成功后，就可以开始玩避障小游戏了。下面效果是使用两个不同游戏 ID 名玩游戏后，对游戏积分排序的结果。

主题 "siot/积分统计"	主题 "siot/积分榜 1"	主题 "siot/积分榜 2"
消息 时时木70 美美奶糖59	主题(topic): siot/积分榜1 数据总数 2 最新数据 时时木: 70 描述 时间 2023/4/18 17:38:47 查看详情 清空数据 删除	主题(topic): siot/积分榜2 数据总数 1 最新数据 美美奶糖: 59 描述 时间 2023/4/18 17:38:47 查看详情 清空数据 删除

注意：当智能终端程序上传成功后，游戏排行服务器端的终端中输出了网络受限 7，可能会导致上传的数据无法在游戏服务器端进行排序，这时需要重新运行游戏排行服务器端的程序。



6. 试一试

上面的 SIoT 游戏排行服务器端程序只实现了对两个游戏积分数据排序，下面，大家就根据上面学习过的知识，实现对两个以上的游戏积分数据排序吧！

提示：两个以上的游戏积分数据排序，与两个游戏积分数据排序的方法一样。



任务三：制作游戏排行可视化界面

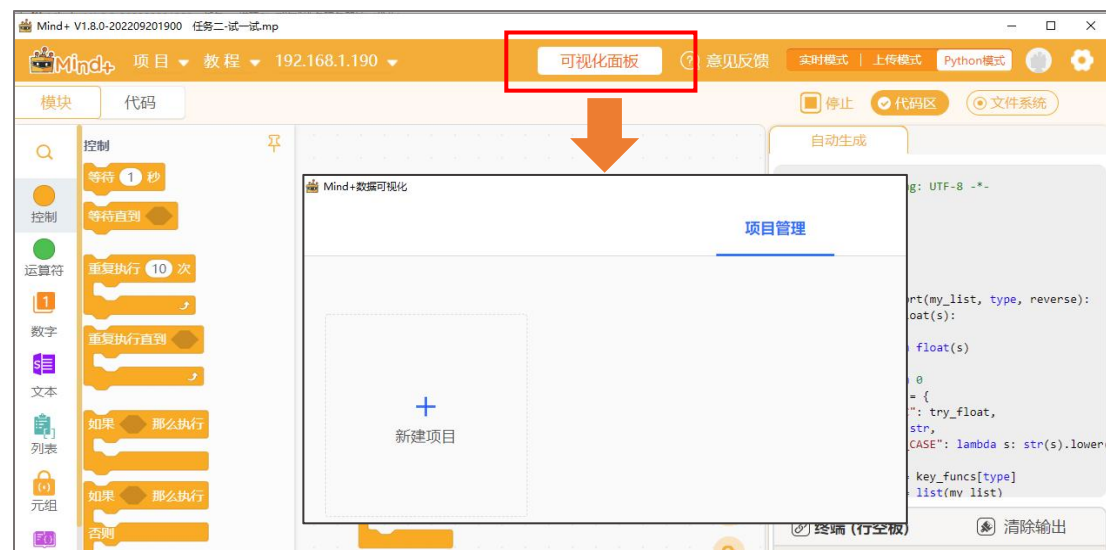
IoT 游戏排行系统的搭建基本完成，为了更好地进行游戏排名的数据呈现，让大家可以看到积分排行情况，使用 Mind+ 的“可视化面板”设计了一个游戏排行的显示界面。

注意：由于，可视化界面是在服务器端程序的基础上进行设计的。因此，这个任务需要在任务二试一试的基础上来完成可视化界面的制作。

1. 制作步骤

新建项目

- (1) 点击任务二 - 试一试编程菜单栏右上方的“可视化面板”按钮，开启“Mind+数据可视化”窗口。



- (2) 鼠标移至“新建项目”处，点击“新建空白项目”；在弹出的窗口中填写好项目名称，点击“确定”项目管理页面就会出现新建的项目标签。



连接 SIoT 服务器

- (1) 新建好项目后，点击对应标签上的“编辑”，进入编辑界面。

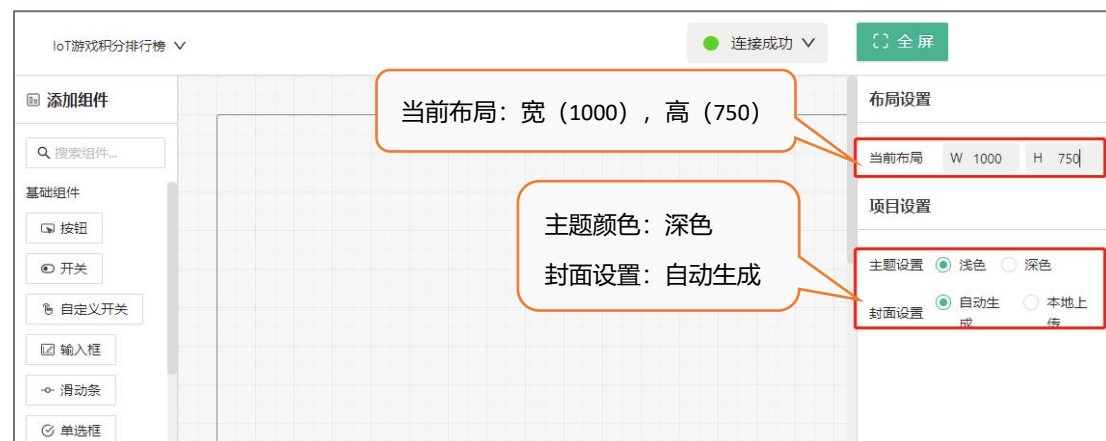


- (2) 然后在页面上方的“菜单栏”，点击“未连接”，选择“数据源设置”，在弹出的窗口中填写要连接的服务器地址“192.168.1.190”，最后点击“完成”，并显示订阅成功！



设计可视化界面

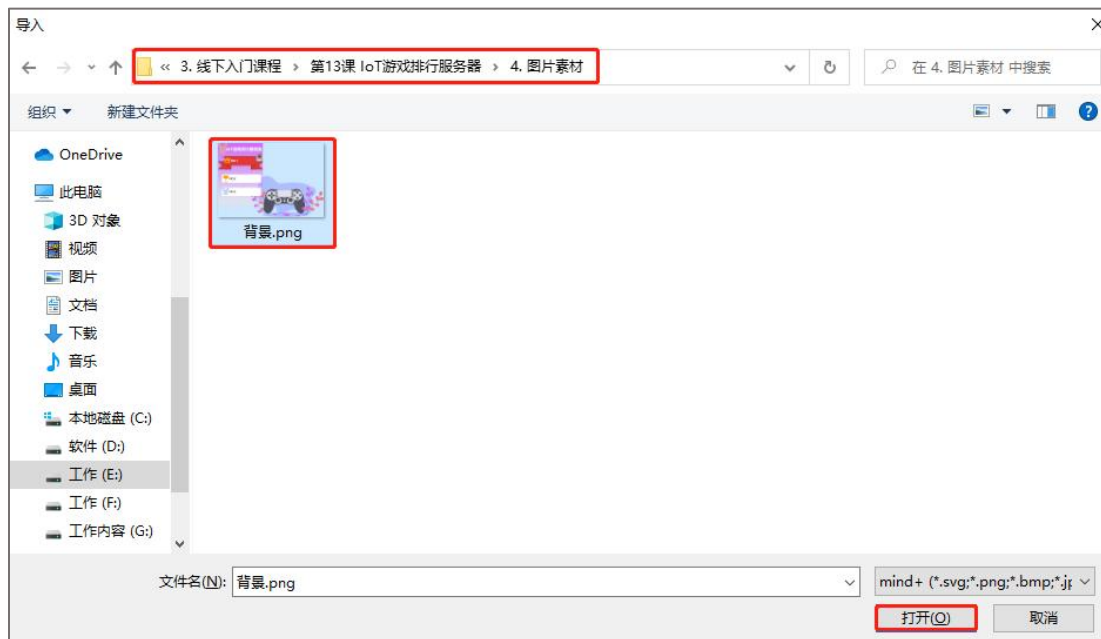
(1) 在可视化编辑页面的右侧“布局设置”以及“项目设置”中，设置画布尺寸、主题、封面。



(2) 在“装饰组件”中，找到“静态图片”，点击搭建区的“静态图片”，在右侧“组件属性”图片中，点击上传。



(3) 为了降低界面设计的难度，直接使用制作好的背景图，在第 13 课的“图片素材”文件中，找到“背景.png”并选择，然后点击“打开”，完成图片上传。



- (4) 图片添加成功后，勾选“隐藏底色”，然后调节静态图片在搭建区显示的大小。调节图片大小有两种方式：第一种，鼠标调节，鼠标点击静态图片后，将鼠标移动至到图片最右下侧，拖动图片调节；第二种，通过“组件属性”中的组件宽度和组件高度来调节。这里设置组件高度设为：820，组件宽度为：575。



- (5) 在“显示组件”中，找到“单行文字”并拖到搭建区，然后在组件区，勾选“隐藏底色”，然后在“topic”中，选择“siot/积分榜 1”。



- (6) 接下来，搭建有同样的方法，在搭建区为 NO.2 与 NO.3 也设置对应的“单行文本”组件吧！需要注意的是，在“组件属性”中，需要对“单行文本”组件设置 Topic 对应的 siot 积分榜。

2. 程序运行

搭建好显示界面后，点击项目名称下拉菜单中的“保存”，然后点击“全屏”查看整个可视化界面。



现在为智能终端的行空板运行避障游戏的程序后，就可以开始玩游戏啦！游戏结束后，IoT 游戏排行服务器端会自动将排序好的积分显示在对应的名次位置。效果如下：



知识园地

1. 了解多设备的组网技术

本节课我们已经学习了多个行空板搭建物联网系统，整个系统中一般包含智能终端、服务器端。智能终端处的行空板，负责收集数据并将数据上传到服务器端，一个物联网系统中可以有多个智能终端；服务

器端的行空板负责接收、存储以及处理数据，最后将处理好的积分数据显示在可视化界面上。

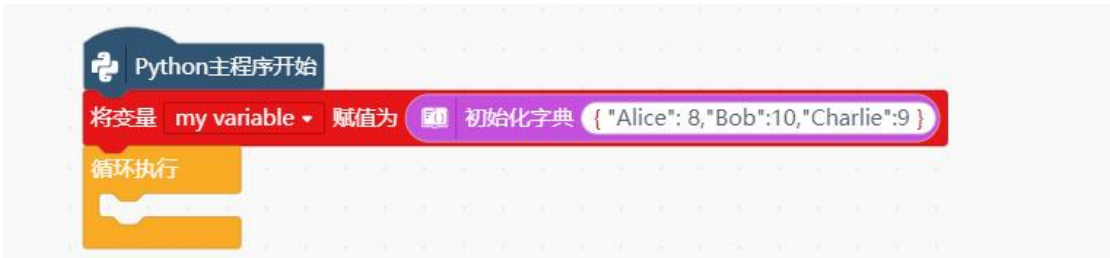


多行空板组建物联网的一般步骤为：配置网络，将所有物联网系统中的设备连入同一网段；开启服务器端 SIoT 服务；SIoT 连接设置，订阅主题，收发数据。当然，如果使用 SIoT2.0，要优先新建主题，才能成功订阅主题。



2. 了解字典的使用方法

Python 中内置了很多种数据结构，例如列表、元组、集合、字典、字符串。而字典是 Python 编程语言中常用的数据结构，用于存储键 - 值对。字典中每个键都必须是唯一的，而每个键可以关联一个值，这样就可以通过键来查找对应的值。例如，下面举了一个字典的例子，它存储了几个人的姓名和年龄。



在这个字典中，“Alice”、“Bob”、“Charlie”是键，它们分别对应的值就是 8、10、9。接下来，我们可以通过键来获取对应的值，例如：



3. 指令学习

Block	功能说明
	该指令用于创建一个字典。
	该指令用于通过字典中的键来获取该键对应的值。
	该指令用于将字典中的所有键或值转换成一个列表。
	该指令用于对列表中的数据，按照列表中的数据类型（数字、字母），进行升序或降序排序。

挑战自我

有些同学会问，如果我们就手上只有一块行空板，那我们是不是就不能实现在 Mind+可视化面板上显示排序后的游戏 ID 和游戏积分呢？如果手上只有一块行空板，我们可以给这块行空板上传任务二 - 试一试（服务器端）的程序，然后进入 SIoT 平台的“siot/积分统计”主题中，使用数据管理中的发送数据功能，来模拟服务器端接收到了对应的游戏 ID 与游戏积分。



下面，大家自己在数据框中输入游戏 ID 与积分，试一试可视化界面，能否正确显示排序后的游戏积分与游戏 ID 吧！

第 14 课 AIoT 班级考勤系统

AIoT 班级考勤系统是一种基于人工智能和物联网技术的考勤管理系统，专门用于学校或培训机构管理学生的出勤情况。它结合了人工智能的面部识别技术和物联网的感知技术，可以快速、准确地识别学生的身份信息 and 出勤状态，并将这些数据自动化地记录在系统中。

接下来，就使用行空板、USB 摄像头、灯带，制作一款这样的 AIoT 班级考勤系统，提高学校管理效率，减轻教师的工作负担，同时也可以加强学生的出勤管理和教育质量控制。



任务目标

智能终端安装在教室门口，用于检测学生到班情况，当智能终端识别到该班级的学生时，屏幕上会显示“打开成功”，灯带显示绿色，然后将该名学生的学号发送给移动终端。移动终端接收到学生打卡成功的信息后，在打卡成功的学生名字后做上标记。

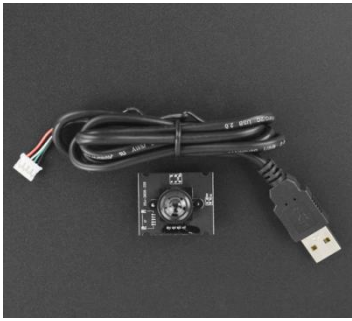
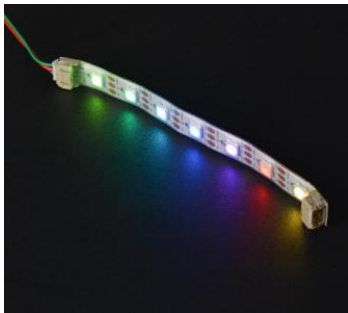


知识点

1. 了解什么是 AIoT
2. 了解什么是人脸识别技术
3. 掌握行空板人脸识别库的使用方法

材料清单

硬件清单:

	
行空板 x2	USB 摄像头 x1
	
RGB 全彩灯带 x1	Type-C USB 线 x1
	
白色硅胶绞线 x1	

软件使用: Mind+编程软件 x1

下载地址: <https://mindplus.cc/>

动手实践

AIoT 班级考勤系统主要是利用人脸识别技术实现对学生人脸进行识别, 人脸识别成功后就将该名学生的 ID 编号上传到移动终端, 移动终端显示签到成功的学生名单。因此, 主要从采集并训练人脸、智能终端是并上传人脸信息、移动终端显示签到名单这三个任务来实现。

任务一: 采集并训练人脸

通过摄像头采集班级学生的人脸图片, 并将它们存入指定文件夹, 建立人脸图片集。然后对人脸图片集进行训练, 获得专用的人脸识别模型。

任务二：智能终端识别并上传人脸信息

利用任务一训练好的人脸识别模型，当摄像头中出现人脸，就对该人脸进行识别。识别成功，灯带显示绿色，并将识别的学生 ID 上传到 SIoT 平台；否则，灯带显示红色。

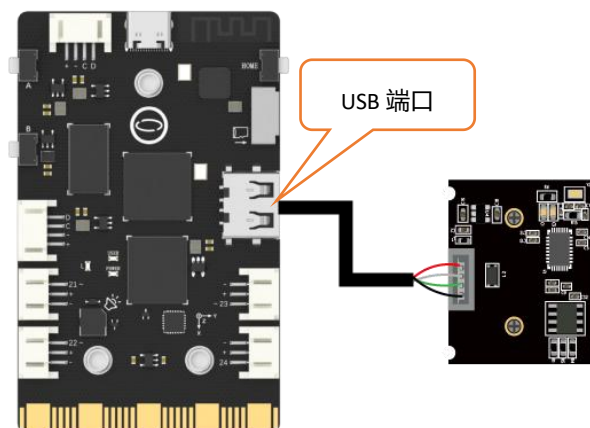
任务三：移动终端显示签到名单

当移动终端接收到识别成功的学生 ID 时，就在 ID 对应的学生签到情况下使用“√”做上签到成功的标记。

任务一：采集并训练人脸

1. 硬件连接

将摄像头连接到行空板的 USB 端口，硬件连接成功后，使用 USB 线将行空板连接到计算机。

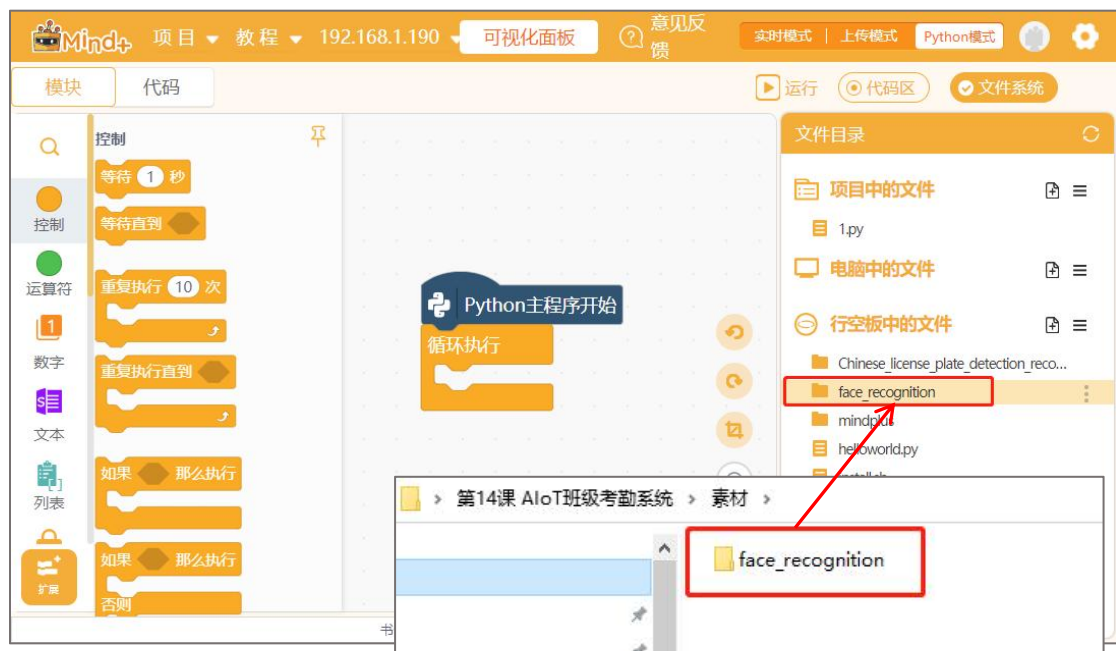


2. 软件准备

(1) 打开 Mind+, 按照下面图示完成软件准备过程。



- (2) 加载人脸识别库：在“素材”文件夹中找到“face_recognition”文件夹，并将该文件夹拖入“行空板中的文件”。



切换至“代码”模式，运行文件夹中的“1-Install_dependency.py”文件并双击，然后点击运行按钮帮助我们离线安装。如果之前安装过 Opencv 的人脸识别扩展库，再次安装时程序会出现警告和报错，但能正常使用。



- (3) 添加人脸识别图形化用户库：点击“扩展”，在“用户库”的搜索栏中输入网址“https://gitee.com/chenqi1233/ext-face_recognition”并检索。



3. 编写程序

- (1) 首先，使用“初始化摄像头直到成功 编号 0”指令，初始化连接在行空板上的摄像头。

注意：指令中的编号 0，是指摄像头编号，默认为 0。



- (2) 采集人脸过程中，当摄像头画面中有人时，才开始采集照片。因此需要使用“导入人脸检测模型”指令，导入人脸检测模型。



- (3) 接下来，使用“等待直到按下按钮 A 保存 (50) 张人脸图片到文件夹 路径 () 人脸 ID 编号 0”指令，完成人脸图片的录入。

注意：保存的人脸图片数量越多，识别就越精准，生成的人脸模型就越久。



- (4) 人脸图片保存成功后，使用“导入人脸图片数据集路径 () 训练人脸识别模型”指令，对保存的人脸图片进行人脸模型的训练。



- (5) 使用“保存训练后的模型文件 路径 ()”指令，对训练好人脸识别模型进行保存。



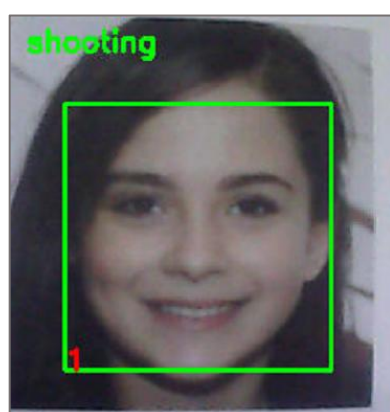
4. 参考程序



5. 程序运行

连接 10.1.2.3, 点击运行, 拿起摄像头对准人脸, 按下 A 键开始录入, 当摄像头检测到人脸后, 行空板屏幕显示 “shooting” 和 ID 编号, 终端记录已保存的图片数量。当终端输出 “人脸识别模型保存成功” 后, 说明人脸图片的采集和训练工作完成。

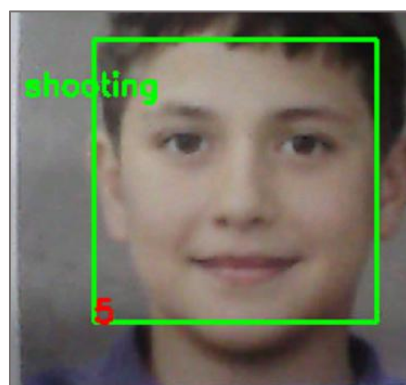
注意: 如果运行出现错误, 请检查摄像头连接情况。录入时, 摄像头最好保持不动, 保证人脸和摄像头的距离适中, 以确保更快地检测到人脸。



6. 试一试

接下来, 大家修改上面程序中的人脸 ID 编号, 完成 5 张人脸图片的录入及人脸识别模型的训练!

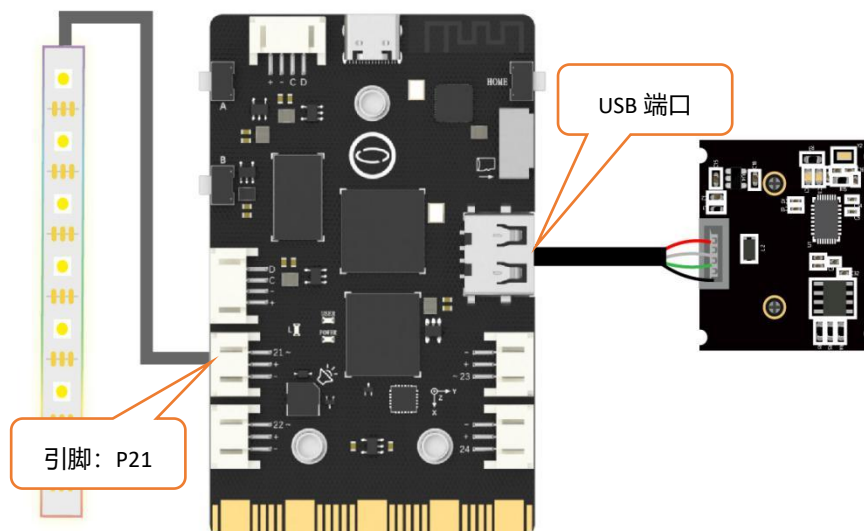
注意: 每次运行程序只能录入一张人脸, 大家记得设置连续的人脸 ID 编号。



任务二：智能终端识别并上传人脸信息

1. 硬件连接

将摄像头连接到行空板的 USB 端口，灯带连接在 P21 引脚，硬件里连接成功后，使用 USB 线将行空板连接到计算机上。



2. 准备工作

- (1) 配置网络：要实现智能终端与智能终端的数据传输，首先要将这两块行空板接入同一网络下，为了可以通过电脑访问 SIoT 服务器，电脑也需要接入同一网络下。

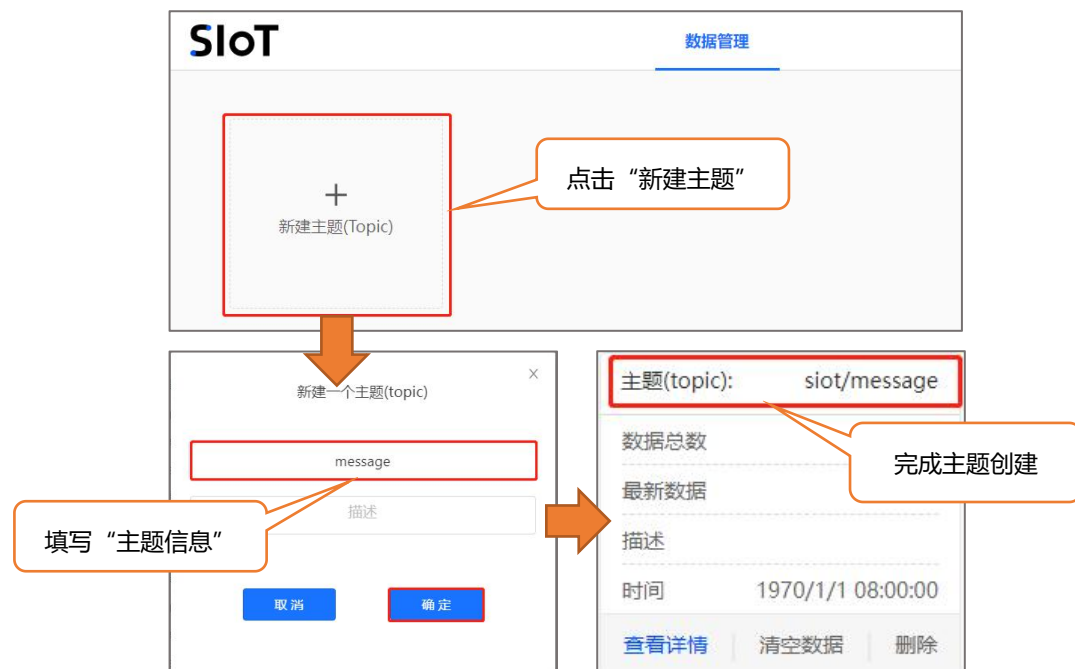
注意：将完成了任务一采集并训练人脸识别模型功能的行空板，作为智能终端。



- (2) 进入 SIoT2.0 物联网平台：浏览器地址栏输入服务器端 IP 地址：192.168.1.190，进入行空板设置界面，进入应用开关，然后点击“打开页面”，进入 SIoT2.0 服务平台登录页面。输入对应的账号和密码，即可完成登录。**账号：**siot，**密码：**dfrobot。

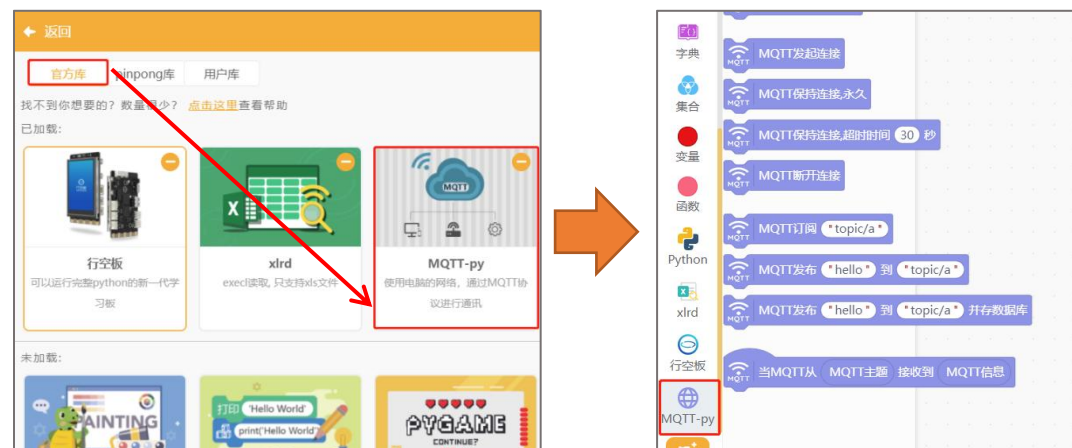


- (3) 创建主题：登录成功后，进入 Slot 数据管理页面。然后点击“新建主题”，在弹出窗口中填写“主题名称”，点击“确定”完成主题的创建，页面会显示对应的主题标签。



3. 软件准备

- (1) 打开 Mind+, 按照任务一完成软件的准备过程。
- (2) 添加人脸识别图形化用户库，按照任务一完成人脸识别图形化用户库的添加。
- (3) 添加 MQTT-py 库，点击“扩展”，在“官方库”中找到“MQTT-py”并点击，完成添加。



- (4) 添加 WS2812 RGB 灯库，点击“扩展”，在 pinpong 库中找到“WS2812 RGB 灯”并点击，完成添加。



4. 编写程序

- (1) 要识别人脸，首先需要使用“初始化摄像头直到成功 编号 0”指令，初始化摄像头。然后使用“导入训练后的模型文件 路径 ()”指令，导入任务一训练好的人脸识别模型。



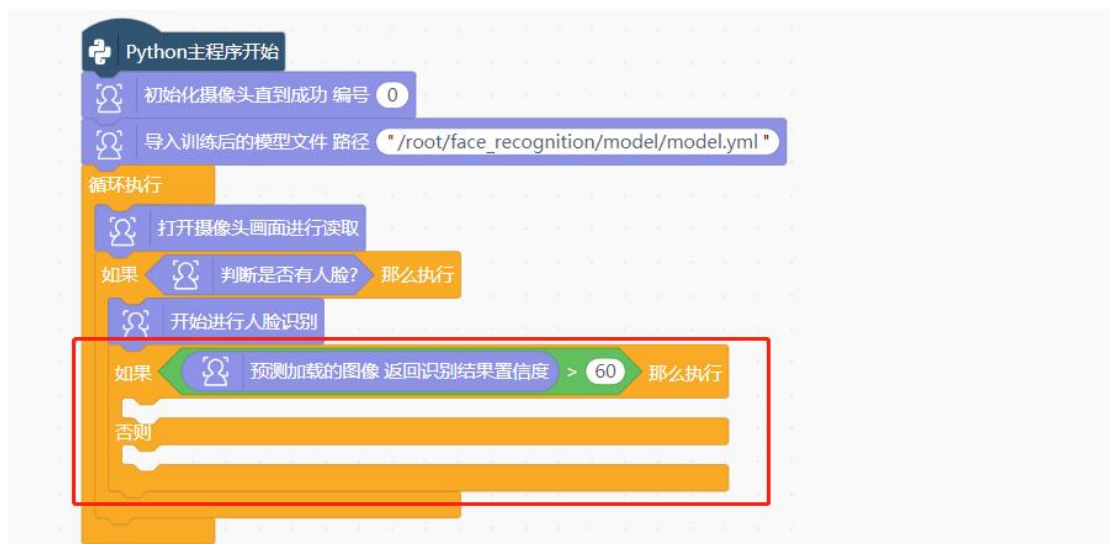
- (2) 人脸识别模型导入成功后，在“循环执行”指令中，使用“打开摄像头画面进行读取”指令，读取摄像头的画面。



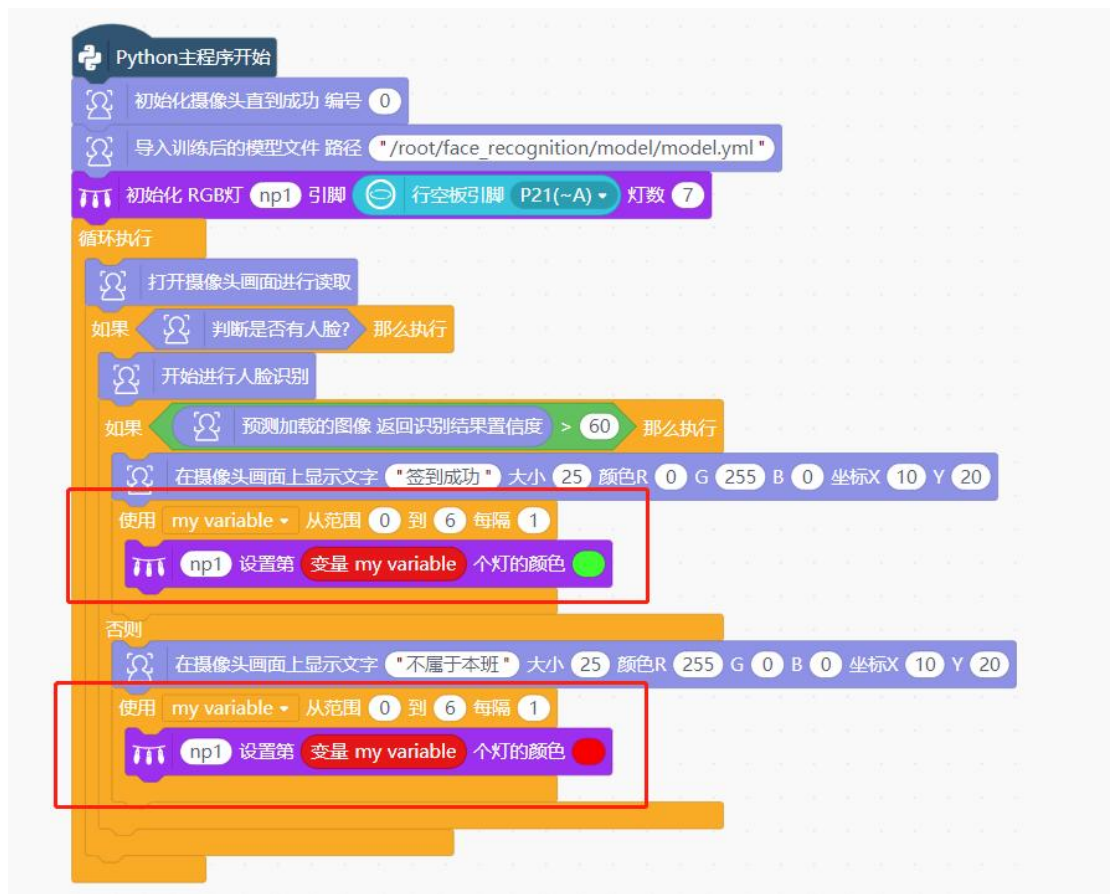
- (3) 接下来，使用“判断是否有人脸？”指令与“条件判断”指令，判断摄像头中的画面是出现人脸。当条件为真时，使用“开始进行人脸识别”指令，对出现的人脸进行识别。



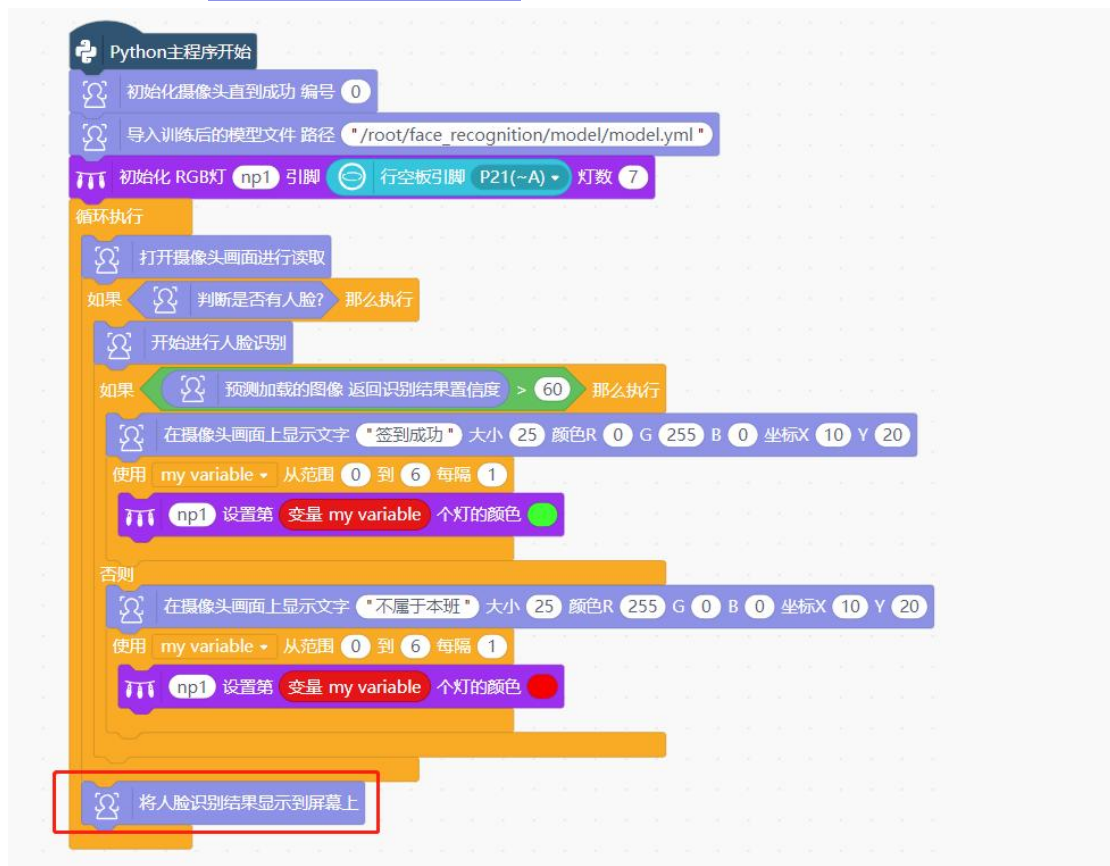
- (4) 为了区分该名同学是否属于本班级，使用“预测加载的图像 返回识别结果置信度”指令，对画面中的人脸进行判断，当该指令返回的结果值越大，说明该名同学是本班学生。这里通过使用“条件判断”指令，判断置信度的返回结果值大于某值（例如 60），来判断进入教室的学生是否属于本班。



- (5) 当判断结果为真时，使用“在摄像头画面上显示文字 () 大小 () 颜色 R () G () B () 坐标 X () Y ()”指令，设置画面上显示文字“签到成功”，并设置灯带的显示颜色为绿色；否则，使用该指令在画面上显示文字“不属于本班”，并设置灯带显示颜色为红色。

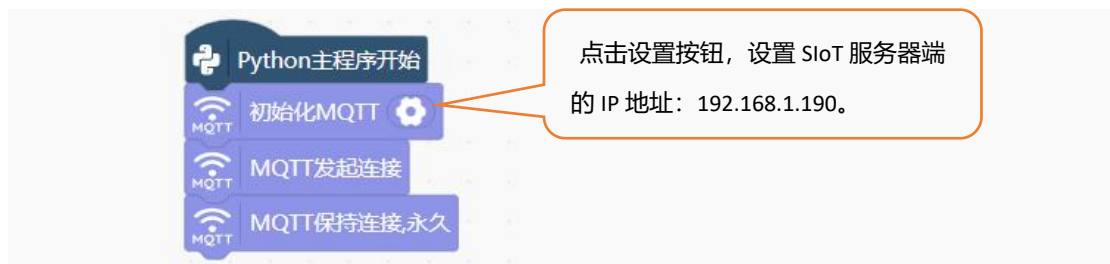


(6) 最后，使用“将人脸识别结果显示到屏幕上”指令，将识别结果显示在行空板上。

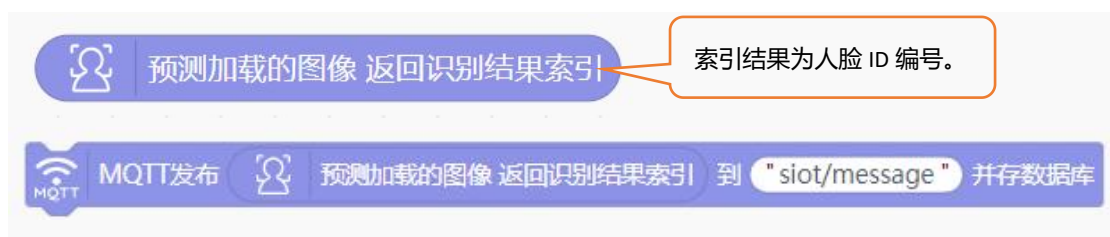


上传人脸编号

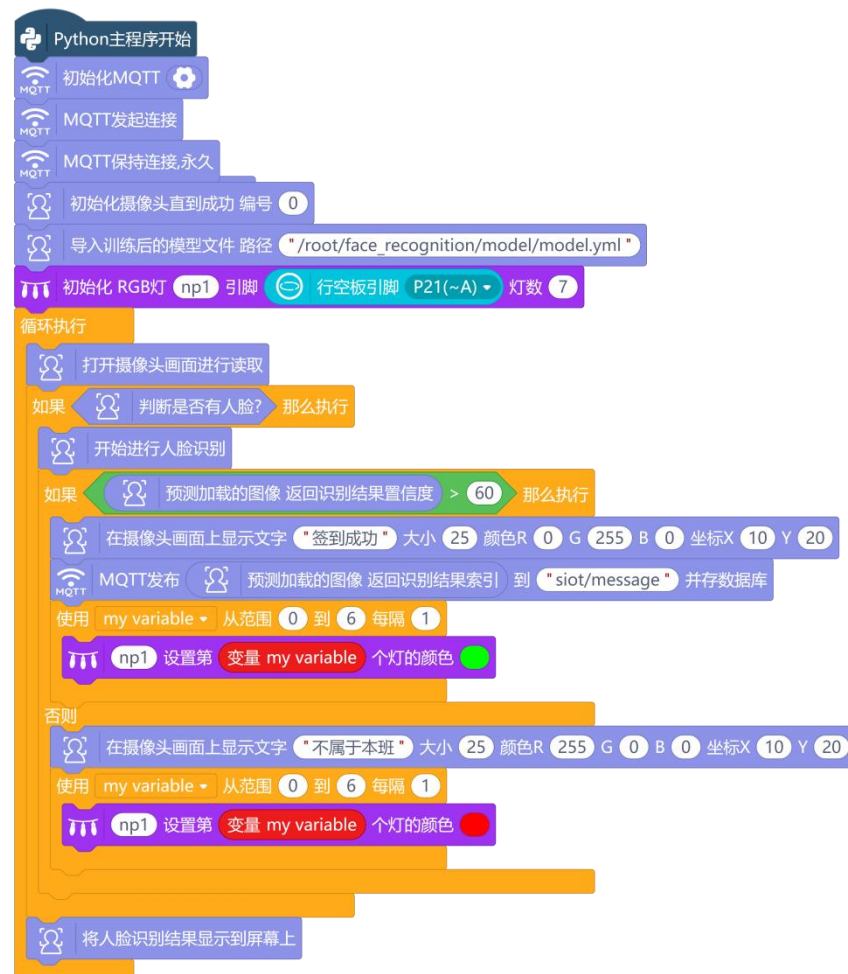
- (1) 要上传人脸编号，在“Python 主程序开始”指令下，使用完成“MQTT 的初始化——发起连接——保持连接”三条指令，完成 MQTT 连接三个步骤。



- (2) 在显示“签到成功”程序下，使用“MQTT 发布 () 到 () 并存入数据库”指令与“预测加载的图像 返回识别结果索引”指令，将识别的人脸 ID 编号发到主题“siot/一班”中。

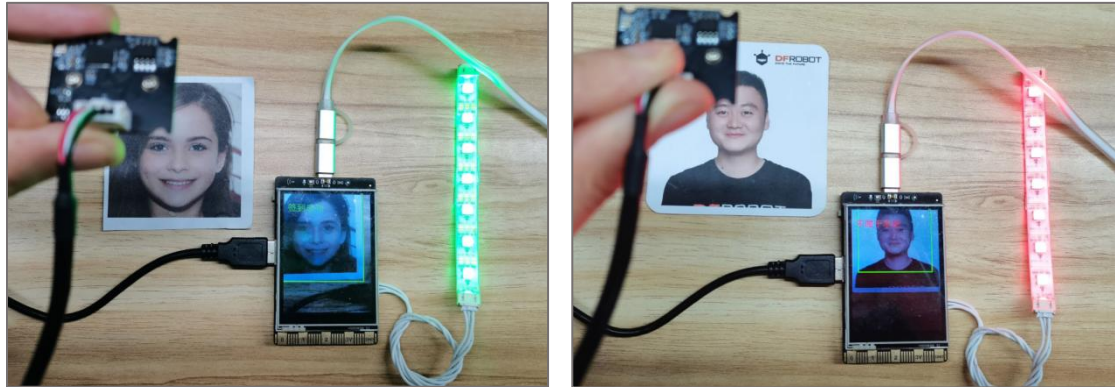


5. 参考程序



6. 程序运行

远程连接 192.168.1.208，连接成功后，点击运行。当摄像头中出现录入过的人脸时，人脸框框选出人脸并显示“签到成功”，灯带显示绿色，然后将人脸 ID 编号发送到主题“siot/message”中；当摄像头中出现的人脸是未录入过的人脸，人脸框框选出人脸并显示“不属于本班”，灯带显示红色。



任务三：移动终端显示签到名单

1. 硬件连接

使用 USB 线将行空板连接到计算机。



2. 软件准备

- (1) 打开 Mind+，按照任务一完成软件的准备过程。
- (2) 添加 MQTT 库，按照任务二中的添加方法完成。

3. 编写程序

开始编写程序之前，观察一下移动终端的界面上都包含了哪些元素？背景图片、签到情况的 ID 编号、时间。



设计 UI 界面

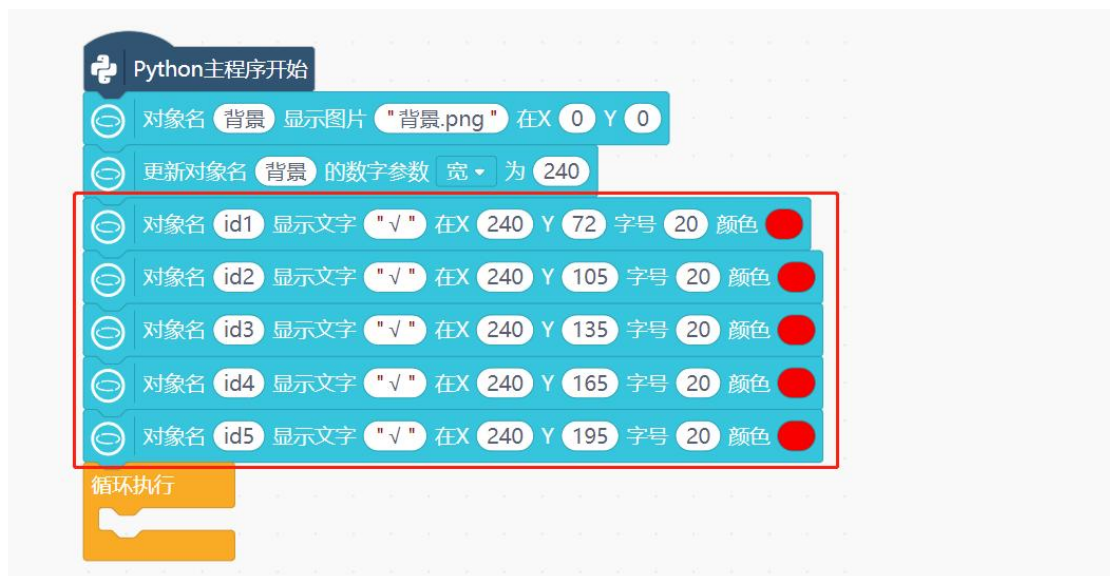
- (1) 为了减小编程难度，将固定的文字及表格制作成了一张背景图。下面，将“图片素材”文件夹中的“背景.png”导入文件系统中。



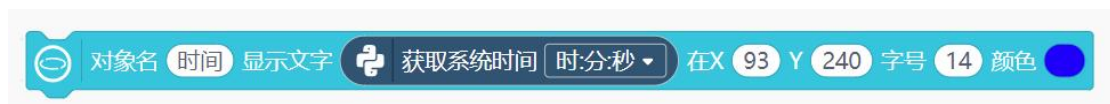
- (2) 在“Python 主程序开始”指令下，使用“显示图片”指令，设置“背景.png”在坐标 (0,0) 处显示。使用“更新对象名 () 的数字参数宽为 ()”指令，设置背景图片的宽为 240。



- (3) 使用“显示文字”，按照分析好的坐标，设置 ID1-ID5 的对象名及坐标，并设置字号为 20，颜色为红色。因为开始界面上签到情况的“√”是隐藏的，因此，修改 ID1-ID5 对象的 X 坐标为 240。



- (4) 使用“显示文字”指令，在坐标（93,240）的位置设置显示文字为“获取系统时间时分秒”，并设置字号为 14，颜色为蓝色。



显示签到成功的学生名单

- (1) 要接收智能终端发送识别成功的人脸信息，首先在“Python 主程序开始”指令下，使用完成“MQTT 的初始化——发起连接——保持连接”三条指令，完成 MQTT 连接三个步骤。



- (2) 使用“MQTT 订阅”指令，订阅主题“siot/message”，只有订阅该主题后，才能接收移动终端发送的信息。



- (3) 在“当MQTT从MQTT主题接收到MQTT信息”指令下，使用“条件判断”指令，判断接收到的学生ID编号。当接收到“MQTT信息”为“1”时，说明学号为1的同学签到成功，使用“更新对象名()的数字参数x为()”指令，更新对象名ID1的x为185。

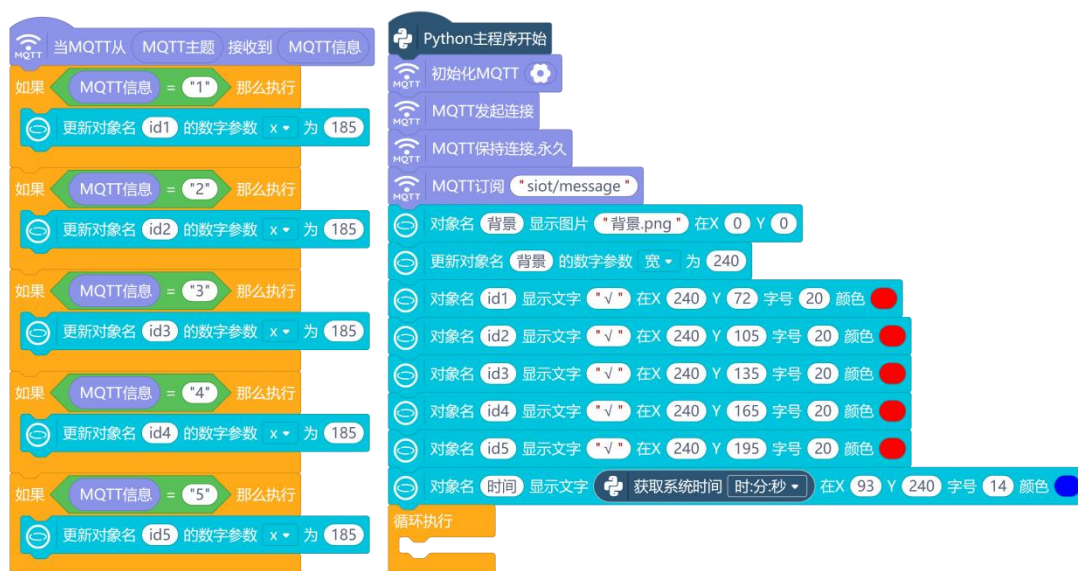


- (4) 当接收到“MQTT信息”为“2”时，说明学号为2的同学签到成功，使用“更新对象名()的数字参数x为()”指令，更新对象名ID2的x为185。



- (5) 下面，大家使用上面的方法，实现当接收到MQTT信息为3、4、5时，更新对应学生的签到情况显示。

4. 参考程序

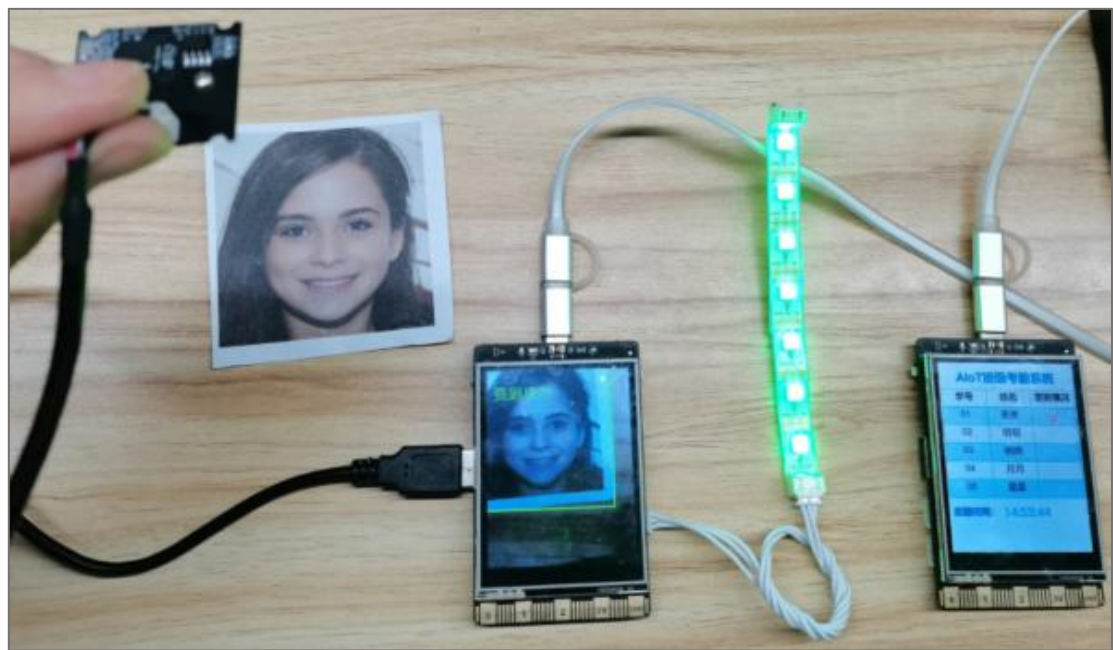


5. 程序运行

远程连接 192.168.1.190，连接成功后，点击运行。行空板上显示学生名单、当前时间以及重置签到表按钮，界面显示如下。

AIoT班级考勤系统		
学号	姓名	签到情况
01	麦麦	
02	明明	
03	晓晓	
04	月月	
05	星星	
当前时间: 14:36:03		

智能终端行空板远程连接 192.168.1.208，运行任务二的程序。程序运行成功后，当智能终端识别到该班级的学生时，将该学生的 ID 编号发送给移动终端，移动终端在对应学号的签到情况下用“√”进行标记。



知识园地

1. 了解什么是 AIoT

AIoT 是 AI 和 IoT 的组合，即人工智能和物联网融合技术。如果说物联网是实现物与物之间的互联互通，

那么 AIoT 相当于在物联网上加入了可以综合分析的“大脑”，它可以借助人工智能实现检测、分析和决策功能，形成基于物联网的智能系统，更好更个性化地提供服务。

一般来说包含机器视觉、语音识别等功能的物联网项目，比如智能安防，都可以称为 AIoT 技术应用。



2. 了解什么是人脸识别技术

第 9 课中，学习了什么是人脸检测技术，这节课我们就来了解一下什么是人脸识别技术。人脸识别，是基于人的脸部特征信息进行身份识别的一种生物识别技术，简单来说，就是对人脸图像进行判断，并识别出其对应人的过程。人脸识别的工作原理和流程和我们日常生活中“认出迎面走过来的朋友”很类似，我们一起来看看。

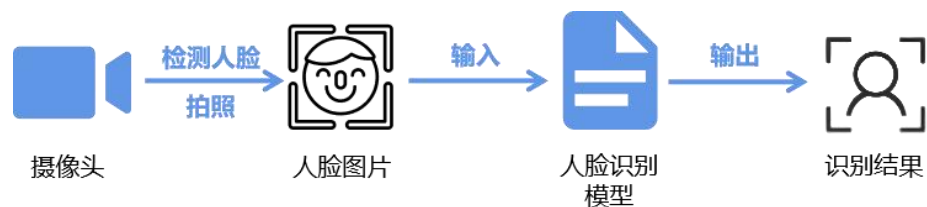
“认出迎面走过来的朋友”可以拆分成两个部分，即我原来认识这个朋友和我认出了她，前者发生在过去，后者发生在现在。在人脸识别技术中，这两个部分分别对应着训练和识别。



总结一下，训练其实就是在帮助机器先去认识这个人。机器并不能像人类的大脑一样可以很容易地记忆人脸图像，它“认识”人脸需要使用一定数量的图片学习并通过特殊的处理形成用于识别的模型。就像在本项目制作中利用摄像头采集人脸图片，然后学习生成人脸模型。它是人脸识别技术的准备工作，也是人脸识别任务的基础。



那么，识别就是让计算机应用人脸模型，去匹配查找检测到的人脸和学习过的人脸哪个最相似，进而获得最终的识别结果。



3. 指令学习

Block	功能说明
初始化摄像头直到成功 编号 0	该指令用于初始化摄像头，默认摄像头设备编号为 0。
导入人脸检测模型	该指令用于导入人脸检测模型。
等待直到按下 按键A 保存 50 张人脸图片到文件夹 路径 */root/face_recognition/picture* 人脸ID编号 0	该指令用于通过行空板 A/B 键，控制摄像头录入人脸图片，并将图片存入路径描述的文件夹。使用时需要写明录入图片数目和当前录入图片的 ID。
导入人脸图片数据集路径 */root/face_recognition/picture/* 训练人脸识别模型	该指令用于读取人脸图片，训练产生人脸识别模型。
保存训练后的模型文件 路径 */root/face_recognition/model/model.yml*	该指令用于将训练好的人脸模型存入指定的路径。
导入训练后的模型文件 路径 */root/face_recognition/model/model.yml*	该指令用于导入指定路径的模型。
打开摄像头画面进行读取	该指令用于程序读取摄像头画面。
判断是否有人脸?	该指令用于检测摄像头画面中是否有人脸。
开始进行人脸识别	该指令用于识别人脸，获取识别结果索引和置信度。
预测加载的图像 返回识别结果索引	该指令用于读取识别到人脸的 ID。
预测加载的图像 返回识别结果置信度	该指令用于读取识别到人脸对应的置信度（可能性）。

 在摄像头画面上显示文字 *id1* 颜色R 50 G 200 B 0 坐标X 10 Y 20	该指令用于在摄像头图像画面上显示文字，使用时可以显示英文、调节颜色。
 将人脸识别结果显示到屏幕上	该指令用于将人脸识别结果显示在行空板屏幕上。

挑战自我

接下来，大家在任务三程序的基础上，添加一个重置签到表的按钮。当这个按钮被点击后，就隐藏签到情况下的标记符“√”。界面显示效果如下：

AIoT班级考勤系统

学号	姓名	签到情况
01	麦麦	√
02	明明	√
03	晓晓	√
04	月月	√
05	星星	√

当前时间：09:41:33

点击重置：

点击重置签到表按钮后，隐藏标记“√”。

AIoT班级考勤系统

学号	姓名	签到情况
01	麦麦	
02	明明	
03	晓晓	
04	月月	
05	星星	

当前时间：15:01:18

点击重置：