第 1 课 声控小夜灯

晚上躺在床上准备睡觉，突然想起客厅的灯还没关。这时候的你，肯定不想再从温暖的被窝里爬出来去按开关。如果只需要拍一下手，灯就自动熄灭，那该多好？反过来，夜里起身喝水或者上厕所时，一片漆黑中摸开关也很不方便。如果拍一下手，灯就自动亮起，过一会儿再自动熄灭，是不是就省心多了？

声控小夜灯就是为解决这些生活小烦恼而设计的。本项目将使用声音传感器检测拍手声，控制 LED 灯自动亮起和熄灭，让你体验用声音控制世界的神奇感觉。

任务目标

设计并制作一个声控小夜灯：当检测到拍手声或足够强的声音时，蓝色 LED 自动点亮，持续 5 秒后自动熄灭；期间再次检测到声音时，LED 灯重新亮起。

学习目标

- 理解模拟输入与数字输出的区别
- 学会使用模拟声音传感器读取环境声音强度
- 学会使用数字 IO 控制 LED 的亮灭
- 掌握阈值判断编程方法

材料清单

硬件清单

行空板 K10(DFR0992) x 1

模拟声音传感器(模拟声音传感器) x 1

数字蓝色 LED 模块 (DFR0021) x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

本项目主要通过下面两个任务，逐步完成声控小夜灯的制作与调试。在任务一中，学会读取声音传感器的数值并通过屏幕观察。在任务二中，利用声音数值触发 LED 亮灭，实现声控夜灯功能。

任务一：获取声音传感器的值

编写程序读取声音传感器输出的模拟值，通过屏幕观察声音大小对应的数值变化。

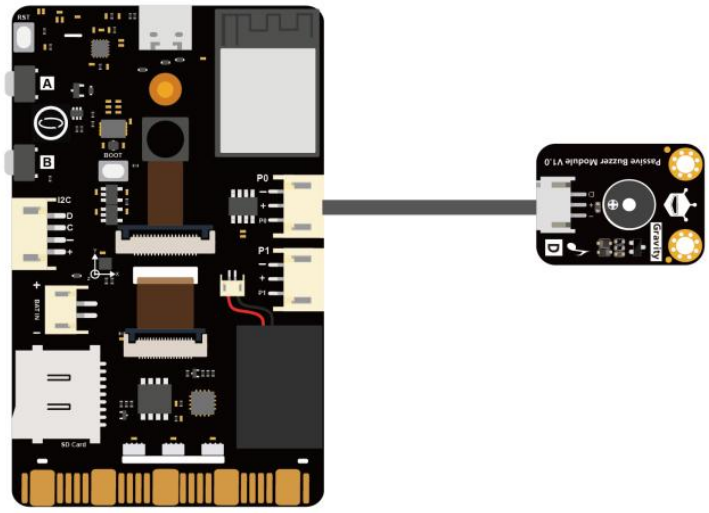
任务二：实现声音触发 LED 亮灭

将声音数值与阈值比较，超过阈值时点亮 LED，延时后自动熄灭。

任务一：获取声音传感器的值

1. 硬件连接

将声音传感器连接在行空板的 P0 引脚。

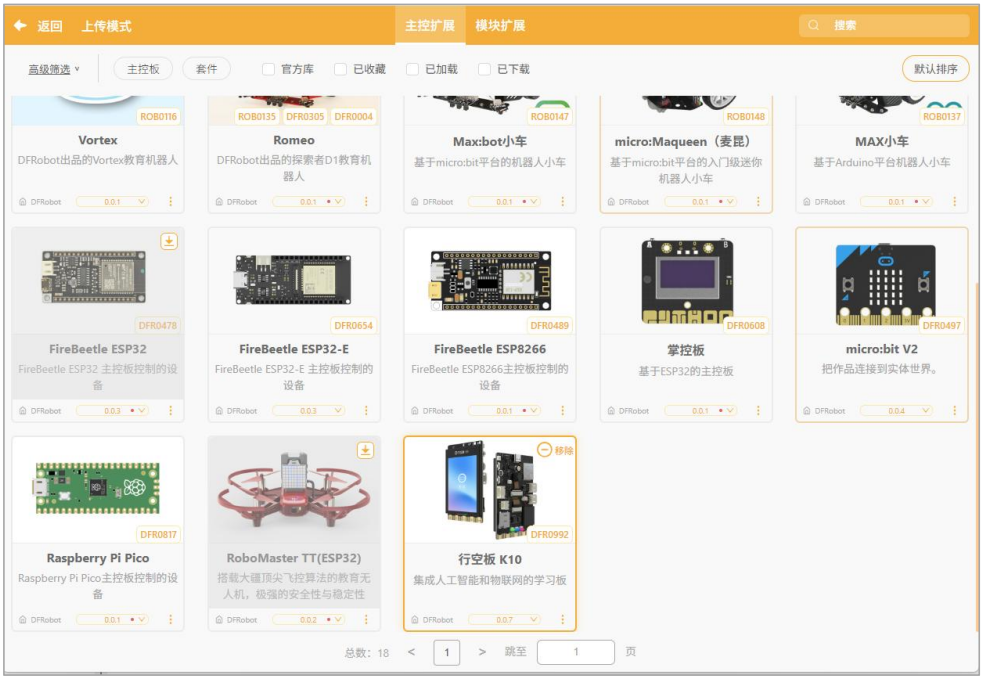


2. 软件准备

1. 打开 Mind+ 软件

2. 选择 上传模式

3. 在"主控扩展"中选择 行空板 K10



3. 程序编写

要获取连接在 P0 引脚上声音传感器检测到值，首先需要使用“设置引脚 模式”指令，设置 P0 引脚为“INPUT”输入模式。

要获取声音传感器检测到的声音值，需要使用“读取模拟引脚 P0”指令。

然后通过“显示文字指令”与“缓存内容显示/显示更新”指令，将声音传感器检测到的模拟值实时的显示在行空板 K10 屏幕上。

4. 程序运行

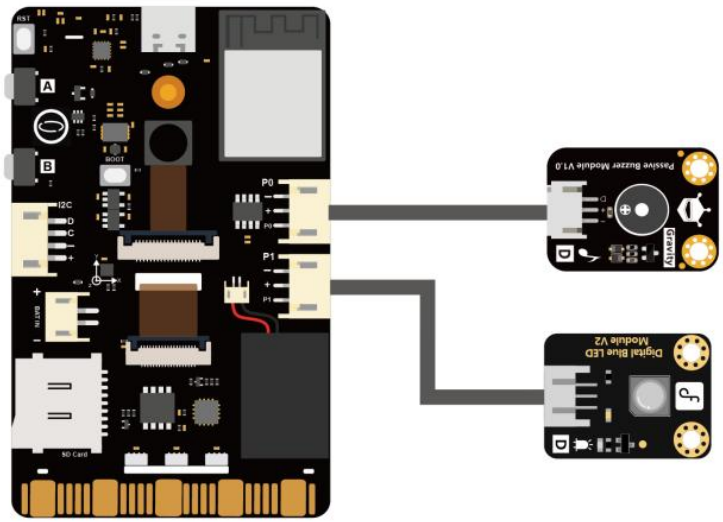
点击右上角"上传"按钮，将程序上传到行空板 K10。观察 K10 屏幕上的声音传感器数值。

尝试拍手、说话、吹气，观察数值如何变化。记下安静时的数值（例如 50~100）和拍手时的数值（例如 200+），为任务二做准备。

任务二：实现声音触发 LED 亮灭

1. 硬件连接

将声音传感器连接在行空板的 P0 引脚，LED 灯模块连接在行空板的 P1 引脚。



2. 程序编写

要控制 P1 引脚的 LED 灯亮灭，首先要使用“设置引脚 模块”，设置 P1 引脚为输出模式“OUTPUT”。

任务一我们已经学会了如何获取声音传感器检测到的声音值，因此直接使用“如果……那么执行……”指令，对“读取模拟引脚 P0”指令进行判断。

当检测到的声音值大于 300 时（检测到拍手或者说话），使用“设置数字引脚 输出”指令，控制 P1 引脚输出高电平（灯亮），并保持 5 秒。

否则，就是人离开后，使用“设置数字引脚 输出”指令，控制 P1 引脚输出低电平（灯熄灭）。完整程序如下：

3. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。拍手或发出声音，观察 LED 是否点

亮。如果太灵敏或没反应，调整阈值后重新上传测试。

知识园地

模拟信号 vs 数字信号

声音传感器和 LED 模块虽然都是通过 3Pin 连接线接到行空板 K10 上，但它们传输的信号类型完全不同。

数字信号就像开关——只有两个状态：要么是 1（高电平，通电），要么是 0（低电平，断电）。LED 模块只认数字信号，所以它只能要么全亮，要么全灭。就像家里的灯开关，按下去灯亮，按回来灯灭。

模拟信号则像一个连续变化的值，可以读出 0 到 4095 之间的任意数值。声音传感器输出的是模拟信号，声音越大数值越高，声音越小数值越低。就像音量旋钮，可以旋到任意位置，不只是开和关。

本项目的有趣之处就在于：读取模拟信号（声音大小），判断后输出数字信号（LED 亮灭），把连续的世界变成干脆的决定。

拓展挑战

1. 延时灭灯：把 “等待 5 秒” 改成 10 秒。

2. 闪烁提醒：LED 点亮后改为闪烁 3 次再熄灭。

3. 双阈值：设置两个阈值——轻声说话（低阈值）LED 微亮，拍手（高阈值）LED 全亮。

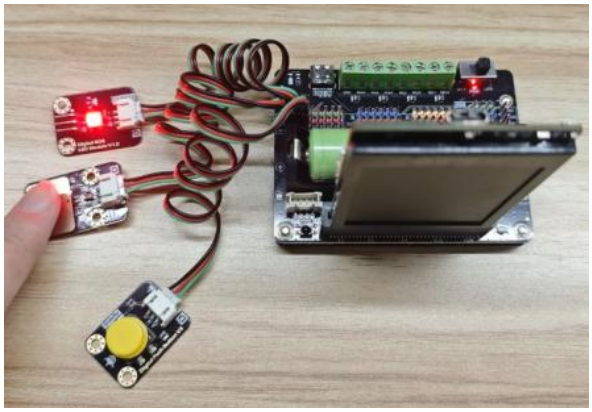
第 2 课 幻彩触控灯

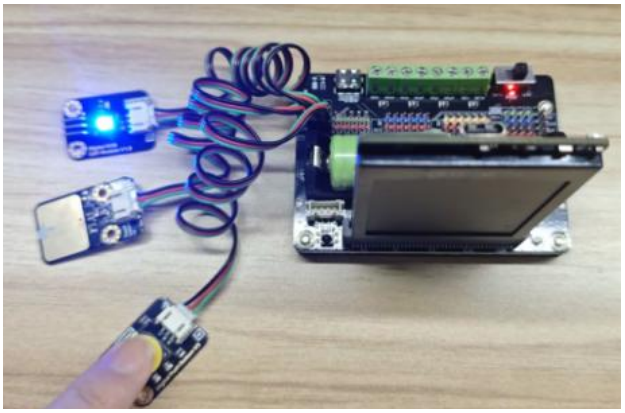
桌面上那盏小台灯，每次都要伸手去够开关，冬天的时候尤其不想把手伸出被窝。如果手指轻轻碰一下，灯就亮了，该多方便？如果心情好的时候还能换个颜色——今天学习用清爽的白光，晚上休息换成温馨的暖黄光，是不是很有意思？

幻彩触控灯就是用触摸代替按压，用色彩调节心情的趣味制作。本项目将使用触摸开关、RGB 全彩 LED 灯和按键模块，制作一个手指轻触就能点亮、按键切换颜色的智能氛围灯。

任务目标

设计并制作一个幻彩触控灯：触摸开关控制 LED 灯的亮灭，按键切换 LED 的显示颜色。





学习目标

- 学会使用数字触摸开关检测触摸信号
- 学会控制 RGB 全彩 LED 灯显示不同颜色

- 学会使用按键模块切换模式
- 了解行空板多功能扩展板 DFR1216 的使用方法

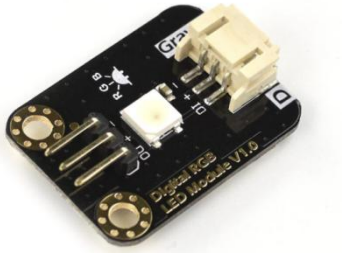
材料清单

硬件清单

行空板 K10(DFR0992) x 1

行空板多功能扩展板 (DFR1216) x 1

数字触摸开关(DFR0030) x 1



RGB 全彩 LED 灯 (DFR0605) x 1

白色按键模块（DFR0029） x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

这个项目，主要是通过下面两个任务，逐步完成幻彩触控灯的制作与调试。在任务一中，实现触摸控制 RGB 灯的亮灭。在任务二中，加入按键切换颜色的功能，让灯光随心变换。

任务一：触摸开关控制 RGB 灯

通过触摸开关控制 RGB LED 的亮灭，触摸一次点亮，再触摸一次熄灭。

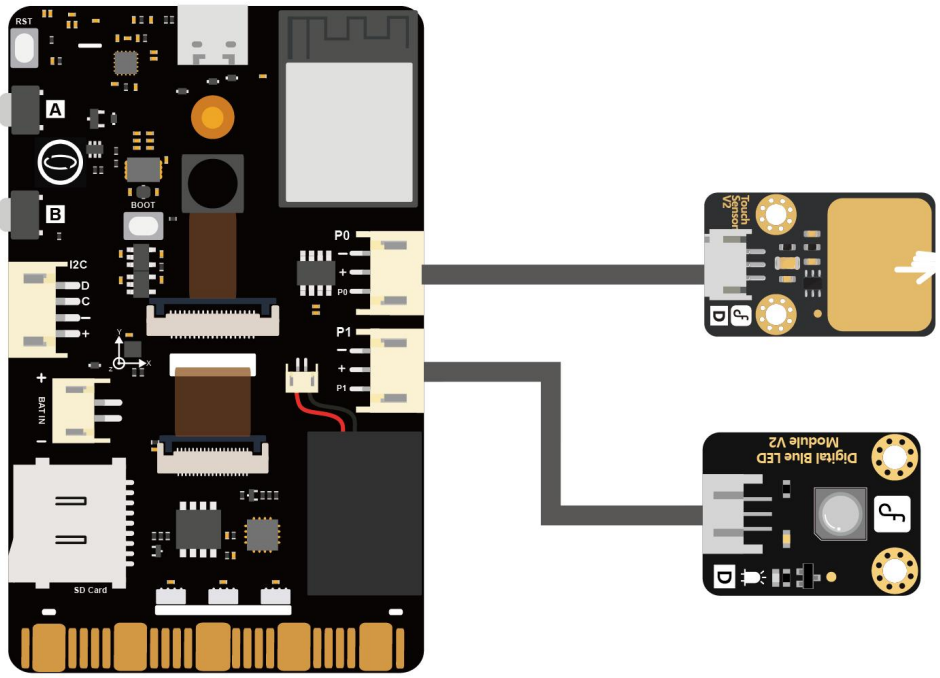
任务二：按键切换颜色

利用按键模块切换 RGB LED 的颜色，每按一次切换到下一种颜色。

任务一：触摸开关控制 RGB 灯

1. 硬件连接

将触摸开关连接到 K10 的 **P0** 端口，RGB 全彩 LED 灯连接到 K10 的 **P1** 端口。



2. 软件准备

1. 打开 **Mind+** 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**
4. 在"模块扩展"的 **显示器** 中搜索 **WS2812**，下载并添加

3. 程序编写

要检测触摸开关是否被触摸，首先需要使用"设置引脚 模式"指令，将 P0 引脚设置为"INPUT"输入模式。

使用"初始化 RGB 灯 引脚 灯总数"指令，初始化连接在 P1 引脚的 RGB 灯模块，设置引脚为 P1，灯总数为 1。

使用"如果……那么执行……否则……"指令，判断触摸开关是否被触摸（读取数字引脚 P0 = 1）。

条件为真时，使用"RGB 灯 引脚 P1 灯号 0 到 1 显示颜色"指令，控制 RGB 灯显示黄色。

否则，使用"RGB 灯 引脚 P1 全部熄灭"指令，控制 RGB 灯关闭。

新建变量 RGB_on 记录灯的状态，触摸一次变量值为 1（亮），再触摸一次变量值为 0（灭），实现触摸切换开关。

4. 程序运行

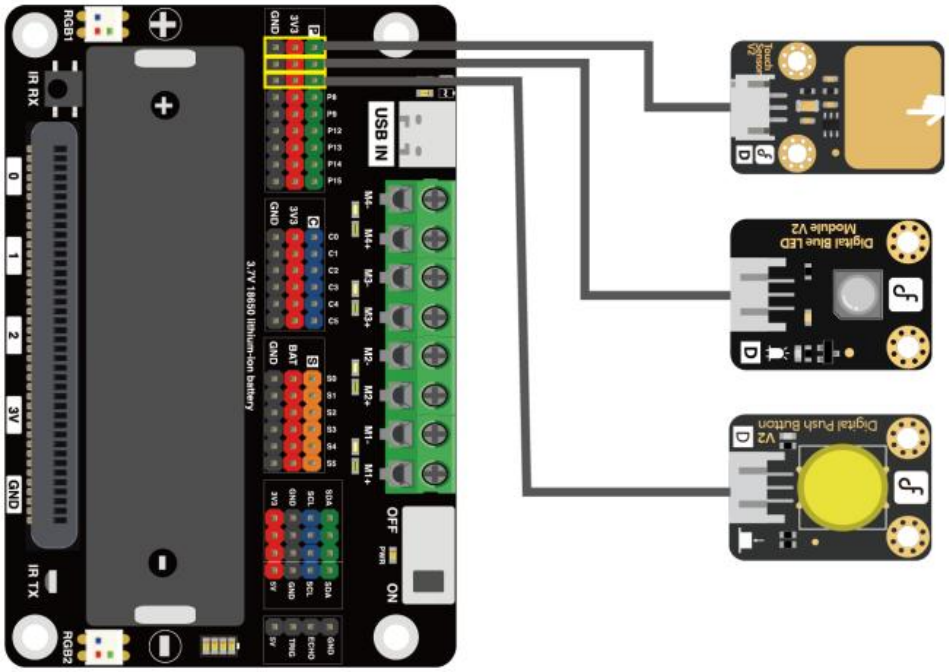
点击右上角"上传"按钮，将程序上传到行空板 K10。用手指触摸开关，观察 RGB 灯是否点

亮（红色）；再次触摸，灯是否熄灭。

任务二：按键切换颜色

1. 硬件连接

将触摸开关连接到扩展板的 P0 端口，RGB 全彩 LED 灯连接到 P1 端口，按键模块连接到 P2 端口。



2. 软件准备

在任务一的基础上，在"模块扩展"中搜索 DFR1216，下载并添加。

3. 程序编写

这个任务是在任务一的基础上，添加通过按钮模块改变 RGB 灯颜色的功能。因此，触摸开关控制 RGB 灯的亮灭功能是需要保留的，我们在任务一的基础上进行添加即可。

创建一个变量 **color_index** 记录当前颜色（1=红，2=绿，3=蓝），初始化该变量的值为 1。

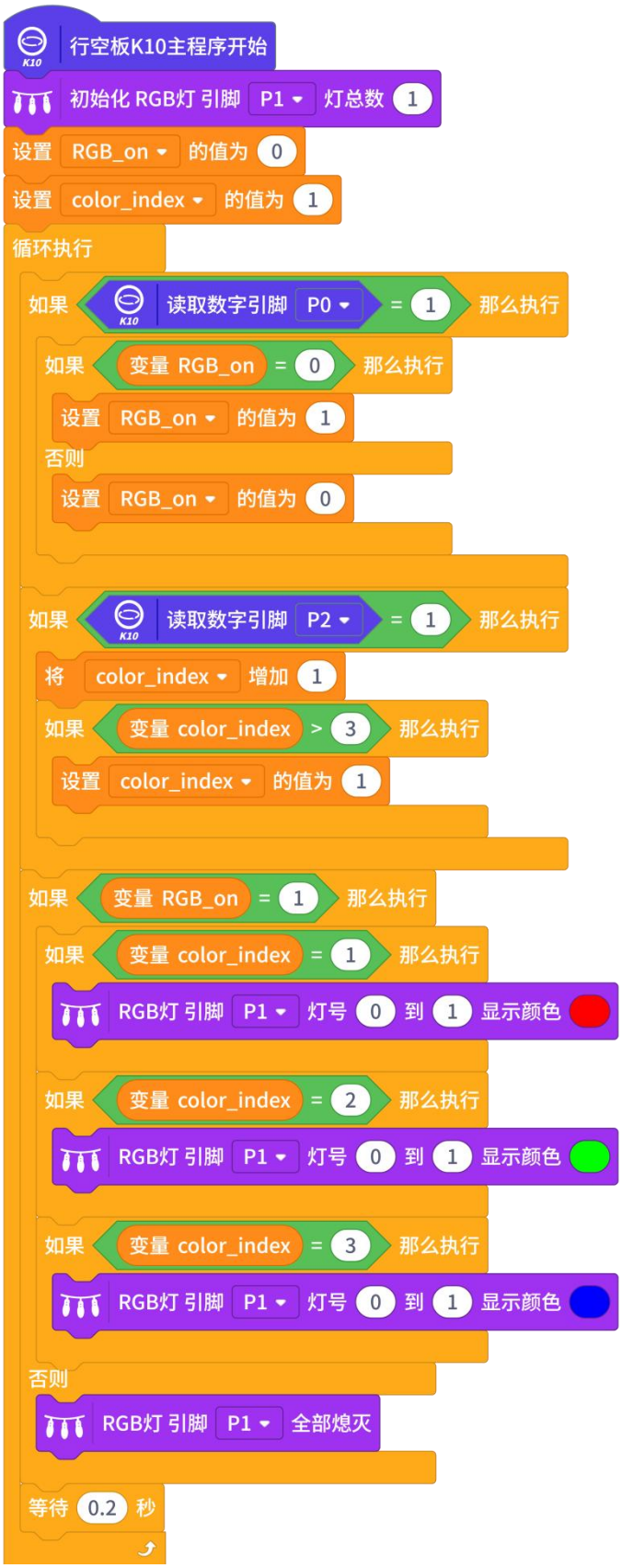
当检测到按钮模块被按下时（读取数字引脚 P2 = 1），使用"将 color_index 增加"指令，让变量 color_index 增加 1。

我们以设置 3 种颜色模式为例，如果变量 color_index > 3 时，使用"设置 color_index 的值为"指令，设置该变量的值为 1。

接下来，就可以在 RGB 灯亮的状态中（变量 RGB_on = 1）控制 RGB 灯进行不同颜色的模式切换了：

- 当变量 color_index 等于 1 时，设置 RGB 灯显示红色。
- 当变量 color_index 等于 2 时，设置 RGB 灯显示绿色。
- 当变量 color_index 等于 3 时，设置 RGB 灯显示蓝色。

完整程序如下：

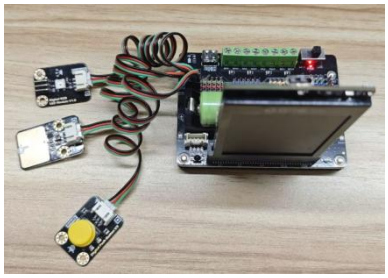
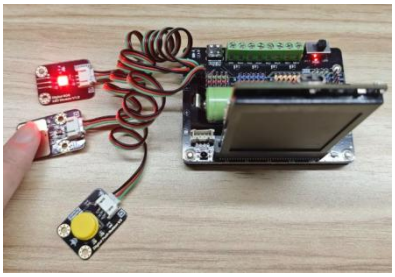
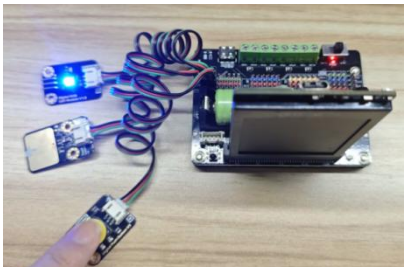


注意：在任务二中，因为所有的传感器都连接在扩展板上，因此没有使用"设置引脚 模式"指令。

4. 程序运行

上传程序后，测试以下操作：

- 触摸开关：控制 LED 亮灭，每次点亮显示当前选中的颜色
- 按键：切换颜色编号，如果 LED 正处于点亮状态则立即更新颜色

知识园地

RGB 颜色混合

RGB 是红（Red）、绿（Green）、蓝（Blue）三种颜色的缩写。这三种颜色是光的三原色，通过不同比例的混合，可以调配出几乎所有的颜色。

- 红色+绿色 = 黄色
- 红色+蓝色 = 紫色
- 绿色+蓝色 = 青色
- 红色+绿色+蓝色 = 白色
- 全部关闭 = 黑色

本项目中的 RGB LED 内部集成了三颗 LED 芯片（红、绿、蓝），通过 WS2812 协议用一根数据线就能独立控制每颗芯片的亮度，从而混合出各种颜色。

行空板多功能扩展板 DFR1216

行空板 K10 本身只有 P0 和 P1 两个 IO 端口，当项目使用的模块超过 2 个时，就需要通过 DFR1216 扩展板来增加可用接口。

接口类型	可用引脚
数字 IO	P0、P1、P2、P8、P9、P12、P13、P14、P15
全功能 IO	C0、C1、C2、C3、C4、C5
舵机接口	S0、S1、S2、S3、S4、S5
电机接口	支持小型直流电机驱动

板载资源	I2C、UART、RGB LED 扩展

将行空板 K10 插在扩展板上即可使用，需安装 3.7V 锂电池供电（将扩展板上的电源开关拨到 ON 端）。

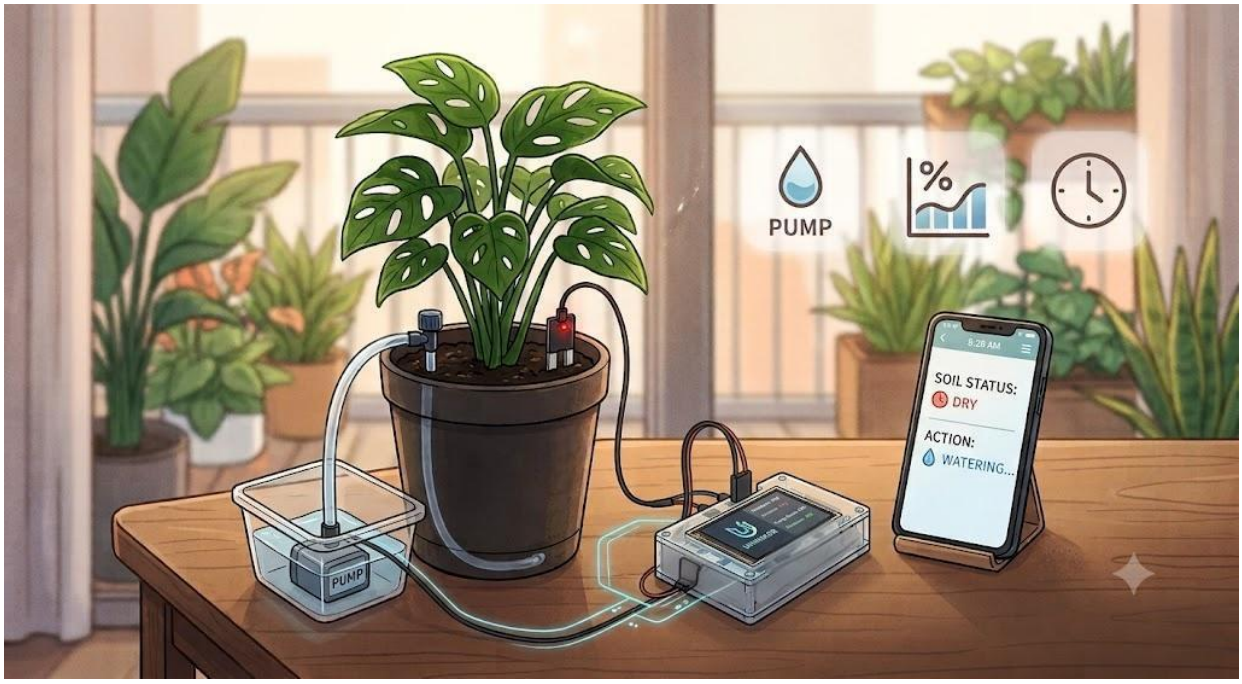
拓展挑战

1. **增加颜色数量：**在颜色表中加入更多颜色（如青色、白色、橙色等）。
2. **长按调光：**按住按键超过 1 秒时进入调光模式，松开后恢复颜色切换。
3. **呼吸灯效果：**点亮 LED 时，让颜色以呼吸（逐渐变亮再逐渐变暗）的方式呈现。

第 3 课 智能灌溉管家

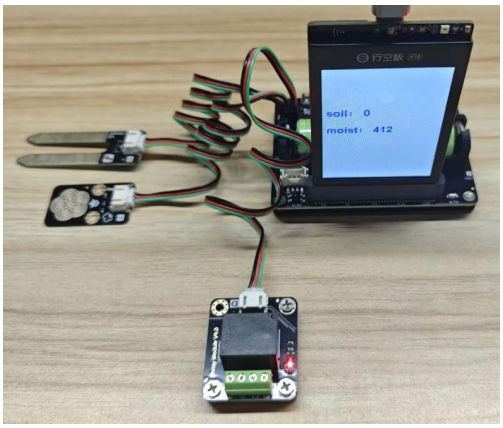
养花最怕什么？出差几天回来，心爱的绿植已经耷拉下了脑袋。每天都浇水吧，有时候忙起来就忘了；不浇水吧，植物渴得慌。要是能有一台设备，能在土壤干燥时自动浇水，该有多省心？

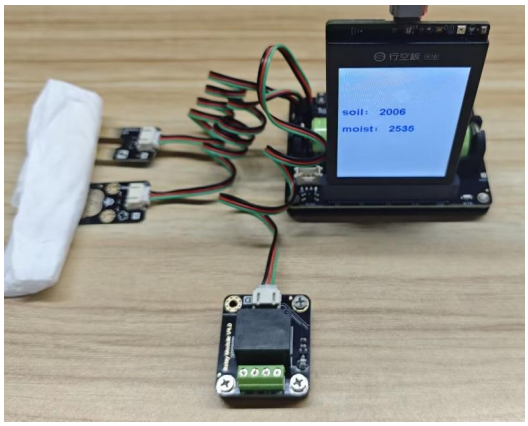
智能灌溉管家就是为了解决这个烦恼而设计的。本项目将使用土壤湿度传感器和水分传感器检测土壤状态，通过继电器控制水泵，实现"干了就浇，湿了就停"的智能浇水功能。



任务目标

设计并制作一个智能灌溉管家：实时检测土壤湿度，当湿度低于设定阈值时自动启动继电器控制水泵浇水。





学习目标

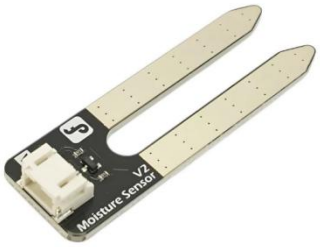
- 学会使用土壤湿度传感器和水分传感器检测土壤湿度
- 理解模拟输入信号的读取与比较
- 学会使用继电器控制外部设备
- 掌握阈值控制的基本方法

材料清单

硬件清单

行空板 K10(DFR0992) x 1

行空板多功能扩展板 (DFR1216) x 1



土壤湿度传感器 (SEN0114) x 1

水分传感器模块（SEN0121) x 1

继电器模块（DFR0017） x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

这个项目，主要是通过下面两个任务，逐步完成智能灌溉管家的制作与调试。在任务一中，

读取两个传感器的湿度值并通过 K10 屏幕观察，了解不同湿度下的数值变化规律。在任务二中，加入继电器模块实现自动浇水控制。

任务一：读取土壤湿度和水分传感器的值

编写程序读取两个传感器输出的模拟值，通过 K10 屏幕观察不同湿度下的数值变化，为任务二设定阈值做准备。

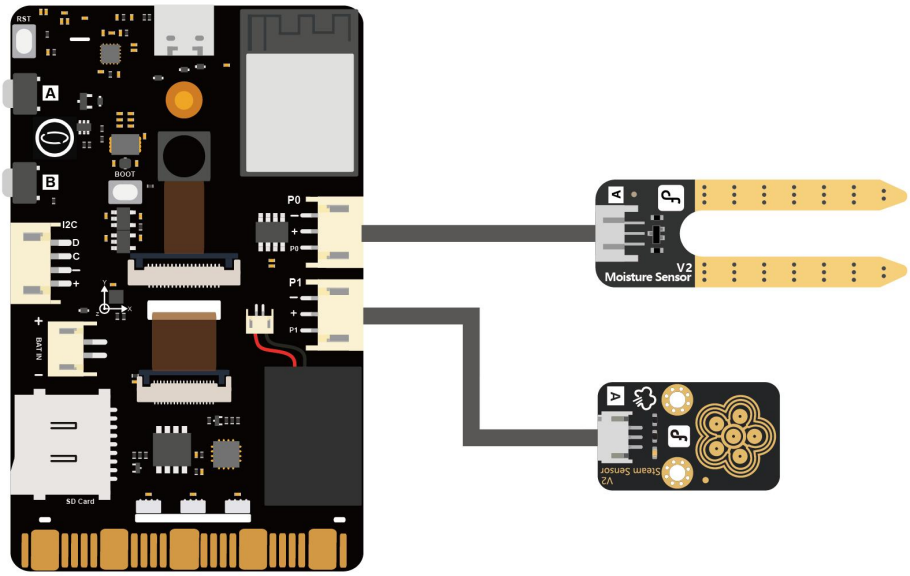
任务二：实现自动浇水控制

将传感器数值与阈值比较，低于阈值时启动继电器，达到阈值后停止。

任务一：读取土壤湿度和水分传感器的值

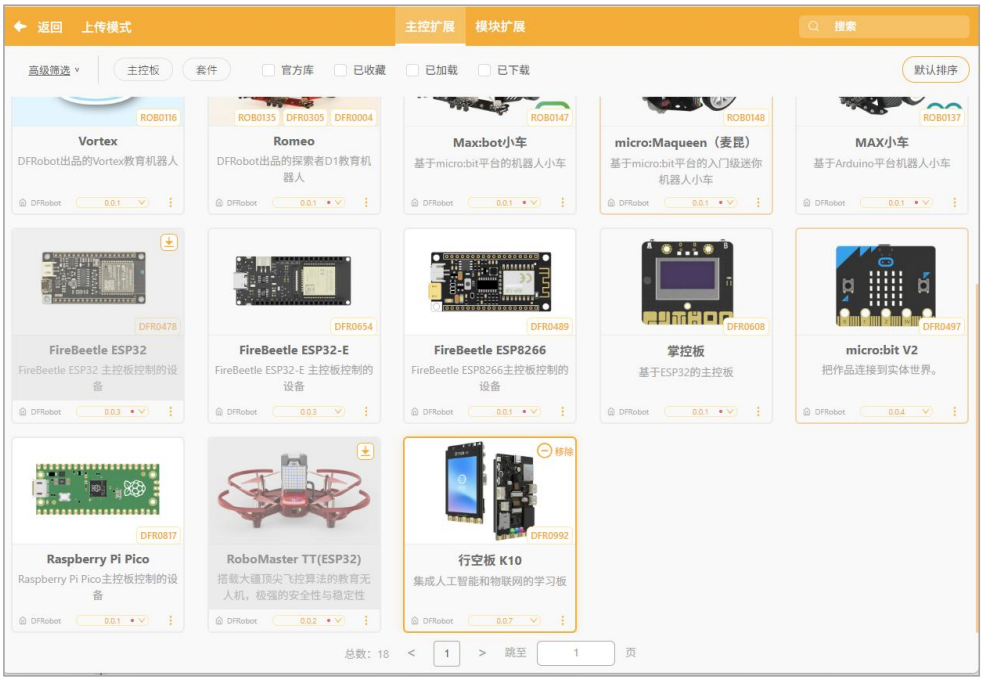
1. 硬件连接

将土壤湿度传感器连接到行空板 K10 的 **P0** 端口，水分传感器连接到行空板 K10 的 **P1** 端口。



2. 软件准备

1. 打开 **Mind+** 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**



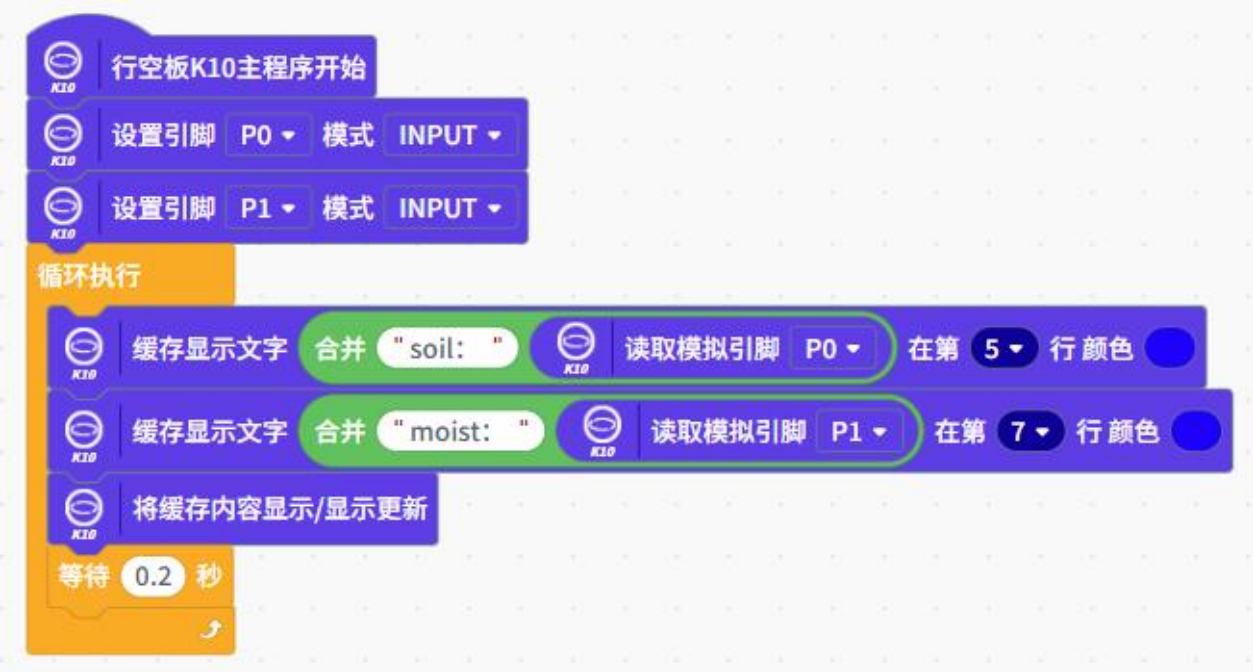
3. 程序编写

首先，需要使用"设置引脚 模式"指令，分别将 P0、P1 引脚设置为"INPUT"输入模式。

然后使用"缓存显示文字"指令，将"读取模拟引脚 P0"指令的返回值也追加到缓存中，对检测到的土壤湿度值进行显示。为了能够区别显示的数值，使用“合并 ”指令，在显示值的前面加上标签“soil： ”。

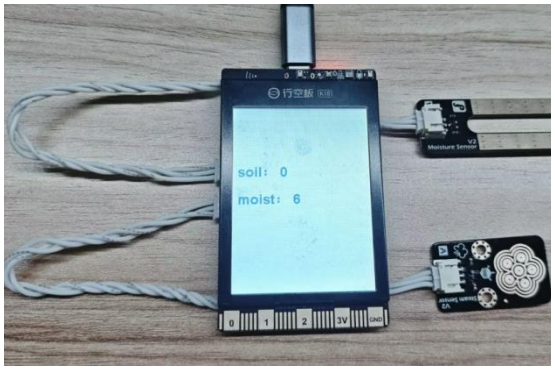
水分传感器的显示也是一样的，只需要修改“读取模拟引脚 ”为 P1，将显示的提示标签修改为“moist： ”。

最后，使用“将缓存内容显示/显示更新”指令，一次性将所有内容显示在行空板 K10 的屏幕上。



4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。观察 K10 屏幕上的传感器数值。



将传感器分别插入干燥土壤和湿润土壤中，记录两种状态下的数值：

状态	土壤湿度传感器 (P0)	水分传感器 (P1)
干燥土壤	数值较低	数值较低

湿润土壤	数值较高	数值较高

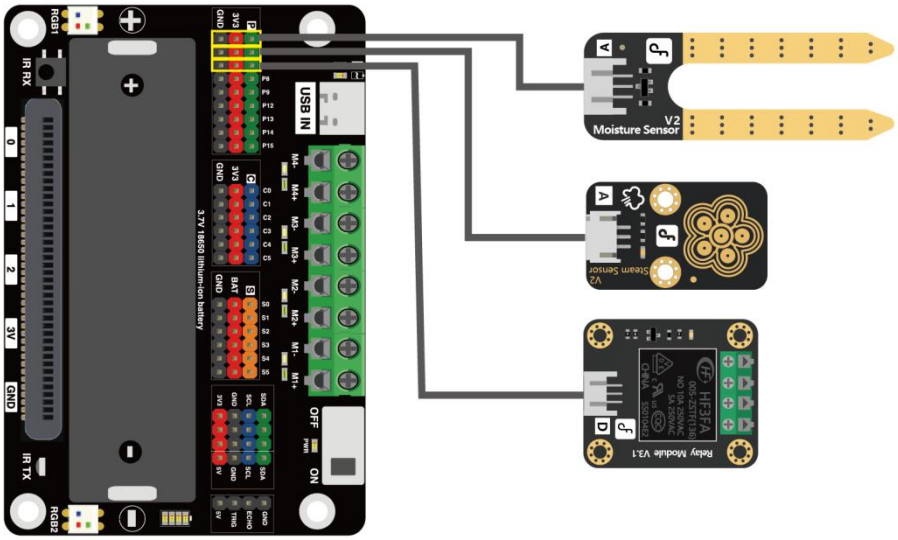
两个传感器都是越湿润数值越高，越干燥数值越低。记下干燥和湿润状态下的两组数值，取中间值作为任务二的阈值。

任务二：实现自动浇水控制

1. 硬件连接

任务二加入了继电器模块，需要使用**行空板多功能扩展板 DFR1216** 来扩展接口。将土壤湿度传感器连接到扩展板的 **C0** 端口，水分传感器连接到扩展板的 **C1** 端口，继电器模块连接到扩展板的 **P2** 端口。

注意：模拟传感器需连接在扩展板的 **C 口**（C0~C5），数字模块连接在 **P 口**（P1、P2、P3、P5、P8、P13、P14、P15）。



2. 软件准备

在任务一的基础上，在"模块扩展"中搜索 **DFR1216**，下载并添加。

3. 程序编写

任务一已经可以获取土壤湿度和水分的检测值。本任务在此基础上，添加自动浇水控制。

创建一个“变量 threshold”，初始化该变量的值为 500，用于设定判断土壤是否干燥的边界值（小于 500，土壤干燥；大于 500，土壤湿润）。

使用"如果……那么执行……否则……"指令进行判断：

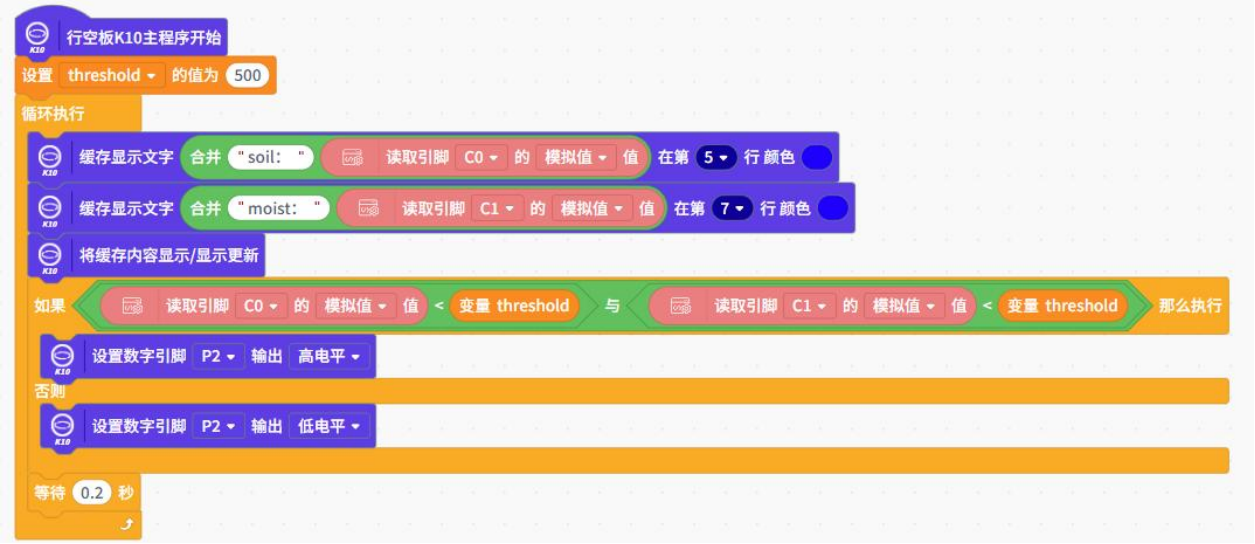
条件：读取模拟引脚 P0 < threshold 且 读取模拟引脚 P1 < threshold（两个传感器检测值都低于阈值）。

注意：使用扩展板时，C 口和 P 口对应不同的库指令——C 口的模拟值需用 **DFR1216 扩展库** 的"读取引脚 模拟值"指令获取，P 口的数字输出使用 **行空板 K10 库** 自带的"设置数字引脚输出"指令即可。

条件为真：说明土壤很干，使用"设置数字引脚输出"指令，将 P2 引脚设为高电平，继电器吸合，开始浇水。

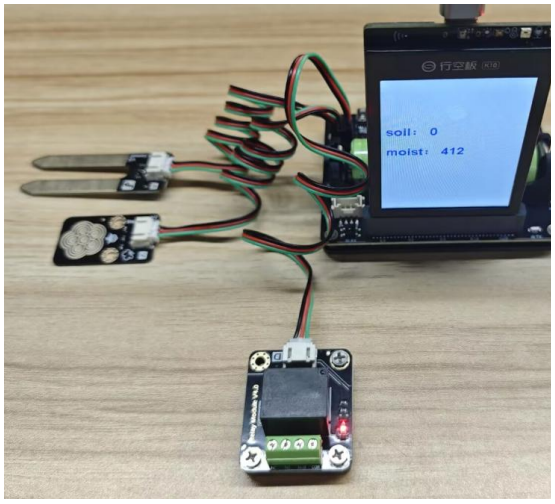
否则：将 P2 引脚设为低电平，继电器断开，停止浇水。

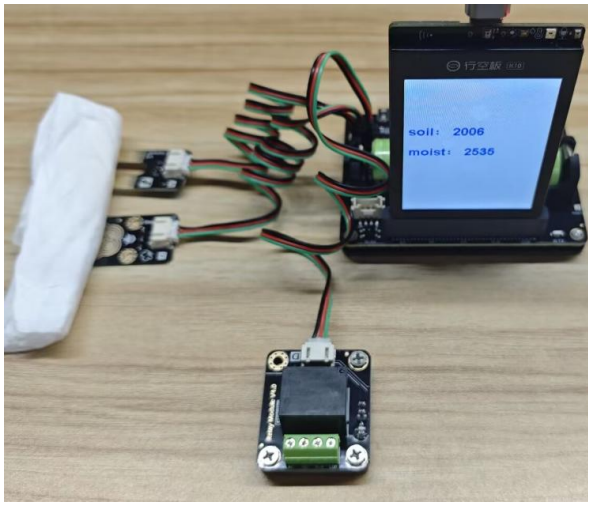
当然，任务一中行空板 K10 屏幕显示所涉及的数据引脚指令，也需同步修改为对应扩展板的模拟引脚 C 口。完整程序如下：



4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。将传感器插入干燥土壤中，观察继电器是否吸合（指示灯亮，听到"咔嗒"声）；再插入湿润土壤中，观察继电器是否释放。





知识园地

土壤湿度传感器 vs 水分传感器

这两个传感器都是通过导电性来检测水分含量。

土壤湿度传感器（SEN0114）利用两根裸露的金属探针测量土壤的导电性。土壤越湿润，探针之间导电性越好，读取的模拟值越高；土壤越干燥，导电性越差，模拟值越低。

水分传感器（SEN0121）同样通过导电性来检测水分，趋势一致——水分越多数值越高。

把两个传感器放在一起使用可以交叉验证，让判断更加可靠。

继电器——小信号控制大设备

继电器就像"一个可以用小力气撬动大石头的支点"。行空板 K10 输出的 3.3V 小信号，通过继电器可以控制 220V 交流电或大功率设备，比如水泵、风扇、灯带等。

它的核心原理是电磁铁：小电流通过线圈产生磁力，吸合内部的金属触点，从而接通或断开大电流电路。

在本项目中，当土壤干燥时，K10 给继电器一个高电平信号，继电器吸合，水泵通电开始浇水；当湿度达标后，K10 输出低电平，继电器断开，水泵停止。这样，行空板 K10 就通过继电器间接控制了大功率的水泵设备。

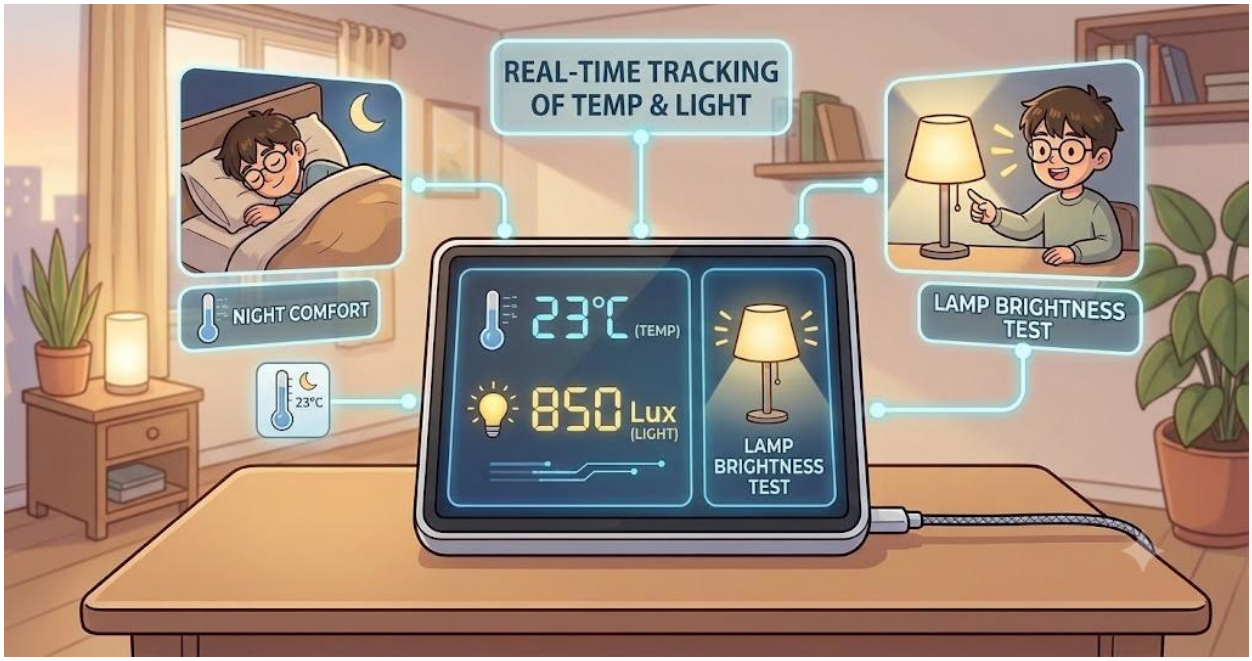
拓展挑战

1. **延时停止**：浇水持续 5 秒后自动停止，防止过度浇水。
2. **手动控制：**增加一个按键，按下后切换为手动模式，此时不受阈值控制，再按一次按键或长按恢复自动模式。

第 4 课 温光环境仪

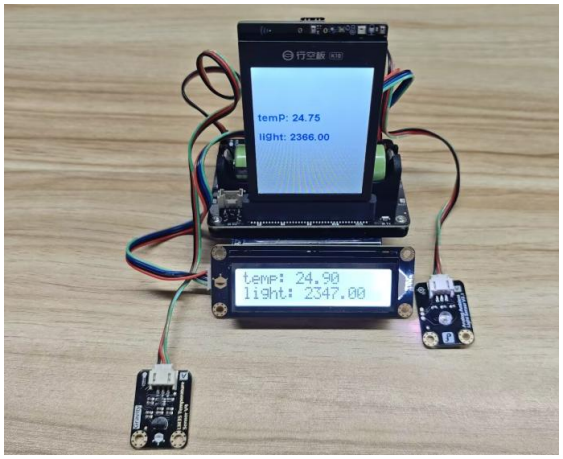
你有没有想过，房间里现在的温度到底是多少？桌上的台灯够不够亮？冬天的时候觉得冷，夏天觉得热，但到底是 30℃还是 35℃，全凭感觉。如果有个小设备能实时告诉你温度和光照强度，是不是很方便？

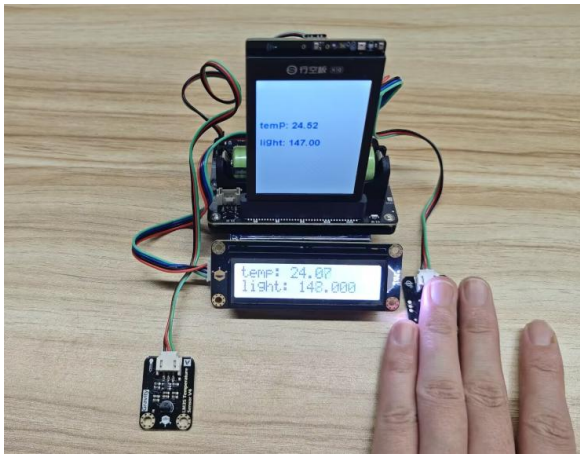
温光环境仪就是这样一个"环境小管家"。本项目将使用温度传感器和光线传感器采集环境数据，并通过 LCD 显示屏实时展示，让你随时掌握身边的温度与光照状况。



任务目标

设计并制作一个温光环境仪：实时采集温度和光照强度，在 LCD 屏幕上显示当前环境数据。





学习目标

- 学会使用 LM35 模拟温度传感器测量温度
- 学会使用模拟环境光线传感器检测光照强度
- 学会使用 I2C LCD1602 RGB 液晶屏显示文字
- 了解 I2C 通信协议的基本概念

材料清单

硬件清单

行空板 K10(DFR0992) x 1

行空板多功能扩展板 (DFR1216) x 1

LM35 模拟线性温度传感器 (DFR0023) x 1

模拟环境光线传感器 （SEN0121）x 1

I2C LCD1602 RGB 背光液晶屏 （DFR0464） x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

这个项目，主要是通过下面两个任务，逐步完成温光环境仪的制作与调试。在任务一中，读

取温度和光线传感器的数值并通过串口观察。在任务二中，加入 LCD 液晶屏，将数据显示在屏幕上。

任务一：读取温度和光线传感器的值

读取两个传感器输出的模拟值，通过串口监视器观察数值随环境变化的规律。

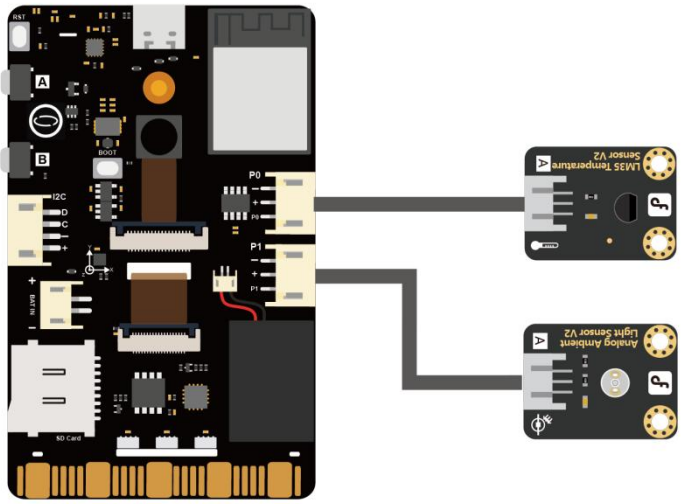
任务二：在 LCD 屏幕上显示环境数据

在任务一的基础上，加入 I2C LCD 液晶屏，将温度和光照数值实时显示在屏幕上。

任务一：读取温度和光线传感器的值

1. 硬件连接

将温度传感器连接到行空板 K10 的 **P0** 端口，光线传感器连接到行空板 K10 的 **P1** 端口。



2. 软件准备

1. 打开 **Mind+** 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**
4. 在"模块扩展"中搜索 **DFR0023**，下载并添加

3. 程序编写

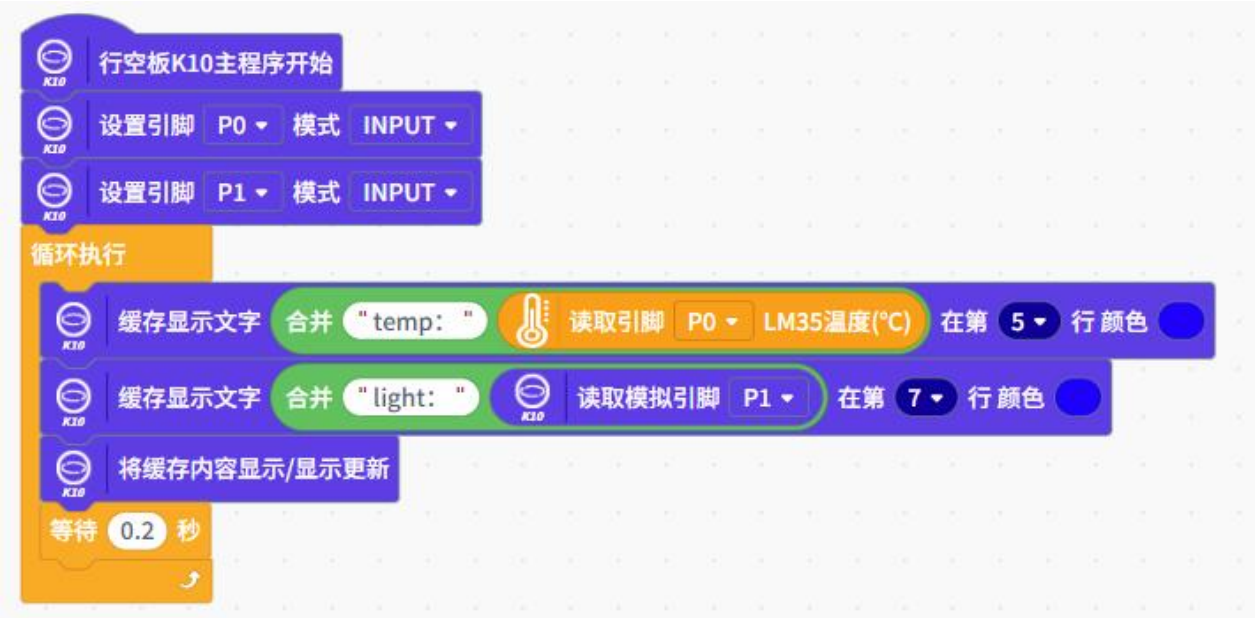
首先，需要使用"设置引脚 模式"指令，分别将 P0、P1 引脚设置为"INPUT"输入模式。

温度传感器使用 DFR0023 扩展库提供的"读取引脚 P0 LM35 温度"指令获取温度值，该指令会自动将模拟值转换为摄氏度。

然后用"合并 "指令在温度值前面加上标签"temp： "，拼成完整的显示文本，再通过"缓存显示文字"指令将其追加到缓存中。

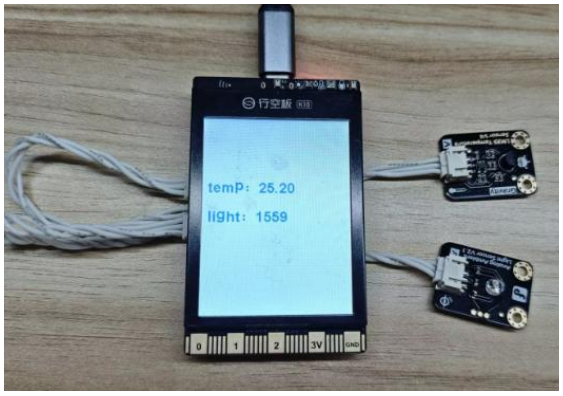
光线传感器的显示也是一样的，使用"读取模拟引脚 P1"指令获取光线值，用"合并"指令加上标签"light： "，再用"缓存显示文字"追加到缓存。

最后，使用“将缓存内容显示/显示更新”指令，一次性将所有内容显示在行空板 K10 的屏幕上。



4. 程序运行

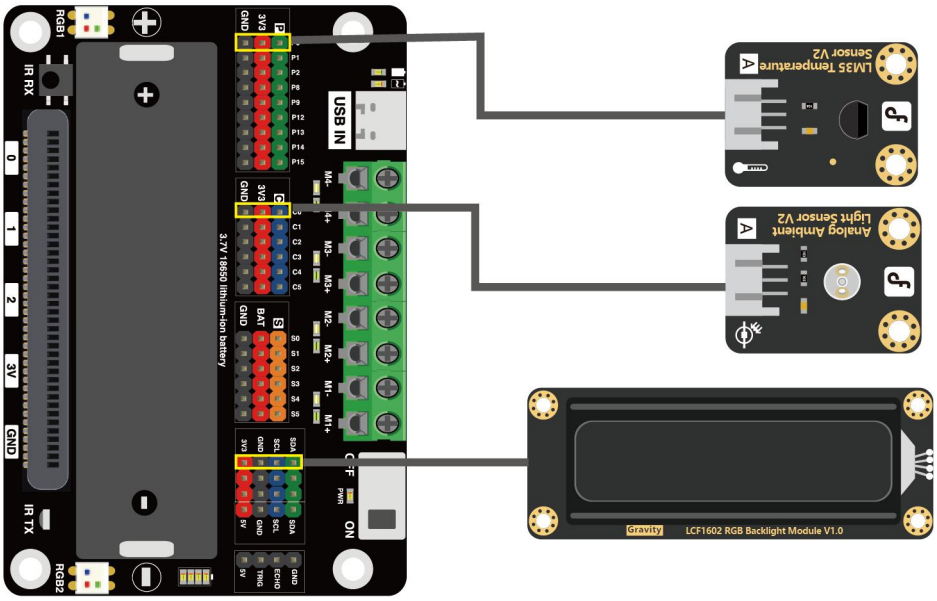
点击右上角"上传"按钮，将程序上传到行空板 K10。观察 K10 屏幕上的温度与光照数值。



任务二：在 LCD 屏幕上显示环境数据

1. 硬件连接

任务二加入了 I2C LCD 液晶屏。温度传感器和光线传感器保持连接在扩展板的 **P0** 和 **C0** 端口，将 I2C LCD 液晶屏连接到扩展板的 **I2C 接口**。



2. 软件准备

在任务一的基础上：

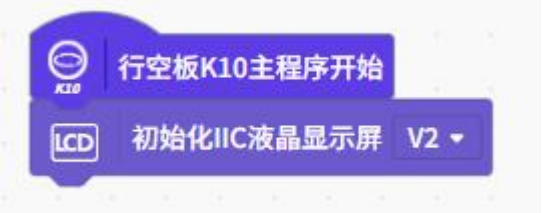
1. 在"模块扩展"中搜索 DFR1216，下载并添加

2. 在"模块扩展"中搜索 DFR0464，下载并添加

3. 程序编写

任务一我们已经成功将温度和光线值显示在行空板 K10 屏幕上，本任务是将获取的温度和光线数据显示在 LCD 屏幕上，显示的内容和任务一样，也是合并标签与数值后的结果。

首先，使用“初始化 LCD1602 RGB”指令对 LCD 进行初始化。

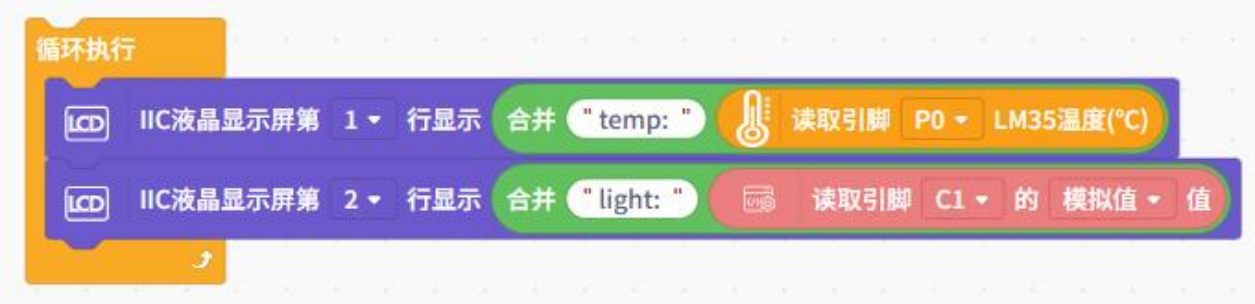


与任务一一样，温度值通过 DFR0023 扩展库的“读取引脚 P0 LM35 温度”指令获取，光线值通过“读取模拟引脚 P1”指令获取。

显示时分两步：

用“合并”指令将标签“temp：”与温度值拼成完整的显示文本，再用“LCD 在第 1 行第 0 列显示”指令显示在 LCD 第一行。

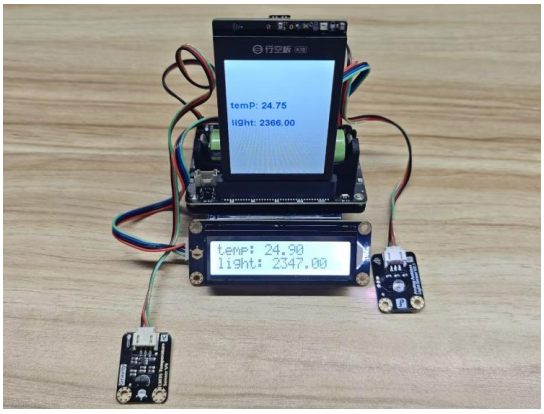
同样的方式，将标签“light：”与光线值拼接后，用“LCD 在第 2 行第 0 列显示”指令显示在第二行。

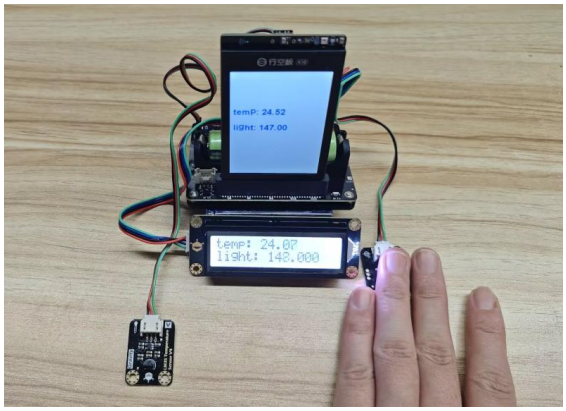


完整程序如下：

4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。观察 LCD 屏幕是否显示温度和光照数值。





知识园地

I2C 通信——多人共用一条线

你有没有想过，这么多传感器，每一根线都要连到主控板的一个端口上，端口不够用了怎么办？I2C（读作"I 方 C"）协议就是为了解决这个问题而设计的。

I2C 只用两根线——数据线（SDA）和时钟线（SCL）——就能让多个设备同时通信。就像班级里老师（主控板）点名，每个学生（传感器）都有自己的学号（I2C 地址）。老师喊"0x2D"，只有学号是 0x2D 的同学会回应。

本项目的 LCD 液晶屏就使用 I2C 接口，它有一个固定的 I2C 地址（0x2D）。通过这两根线，行空板 K10 就能告诉 LCD 屏幕上要显示什么内容，还可以控制背光的颜色和亮度。

LM35 温度传感器

LM35 是一个经典的模拟温度传感器，它的输出电压与摄氏温度呈线性关系：温度每升高 1℃，输出电压就增加 10mV。0℃时输出 0V，25℃时输出 0.25V，100℃时输出 1.0V。

本课使用的 DFR0023 扩展库提供了"读取引脚 P0 LM35 温度"指令，可以直接获取转换后的摄氏度数值，无需手动计算。

拓展挑战

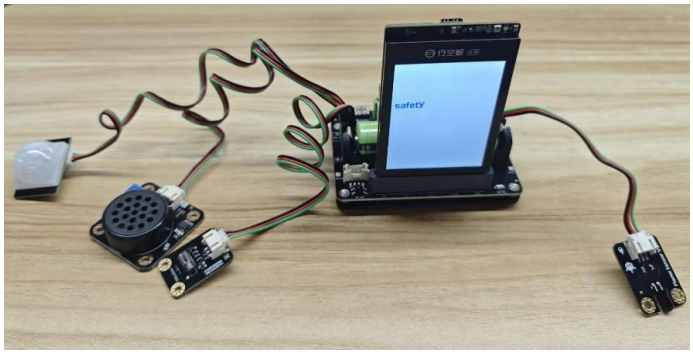
1. **背光随温变色**：根据温度值改变 LCD 背光颜色，低温显示蓝色，高温显示红色，让环境状态一目了然。
2. **温光阈值报警**：设定温度或光照的上下限阈值，超出范围时 LED 模块开始闪烁，起到警示作用。

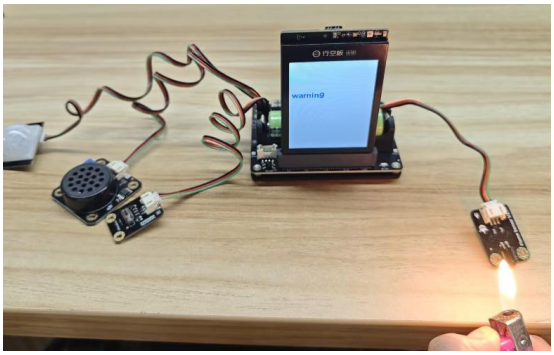
第 5 课 三防安全卫士

在日常生活中，我们经常需要了解周围环境的变化——检测人体入侵或火焰时声光震动报警。本项目将使用人体热释电红外传感器、火焰传感器、带功放喇叭模块、微型振动模块，打造一个智能化的三防安全卫士，让科技服务生活。

任务目标

设计并制作一个三防安全卫士：实时检测人体入侵和火焰，触发声音和震动双重报警。





学习目标

- 学会使用人体热释电红外传感器（SEN0018）
- 学会使用火焰传感器（DFR0076）
- 学会使用带功放喇叭模块（FIT0449）
- 学会使用微型振动模块（DFR0440）

- 了解行空板 K10 板载屏幕的使用方法

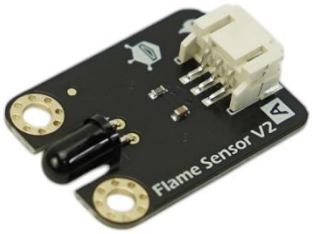
材料清单

硬件清单

行空板 K10(DFR0992) x 1

行空板多功能扩展板 (DFR1216) x 1

人体热释电红外传感器 (SEN0018) x 1



火焰传感器（DFR0076) x 1

带功放喇叭模块（FIT0449） x 1

微型振动模块（DFR0440） x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

这个项目，主要是通过下面两个任务，逐步完成三防安全卫士的制作与调试。在任务一中，先用人体热释电红外传感器和火焰传感器直连 K10 读取数值。在任务二中，使用 DFR1216 扩展板加入其余模块实现完整功能。

任务一：读取人体热释电红外传感器和火焰传感器的值

将热释电红外传感器和火焰传感器检测到的数据实时显示在行空板 K10 屏幕上。

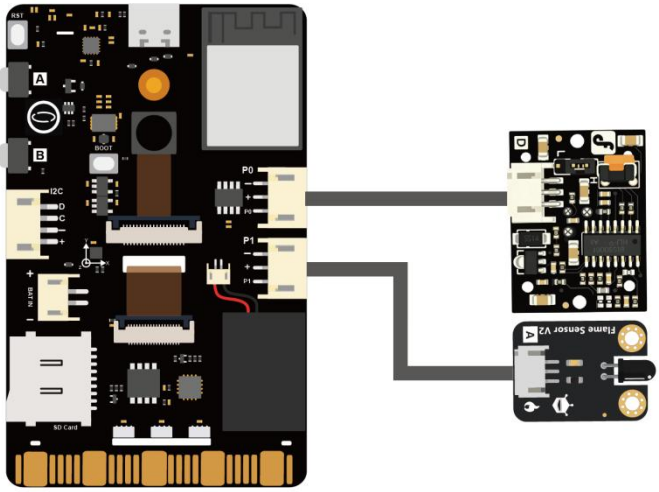
任务二：加入喇叭和振动模块，实现三防报警功能

在任务一的基础上，加入带功放喇叭模块和微型振动模块，编写判断逻辑：当检测到人体入侵 **或** 火焰时，自动触发震动报警。

任务一：读取人体热释电红外传感器和火焰传感器的值

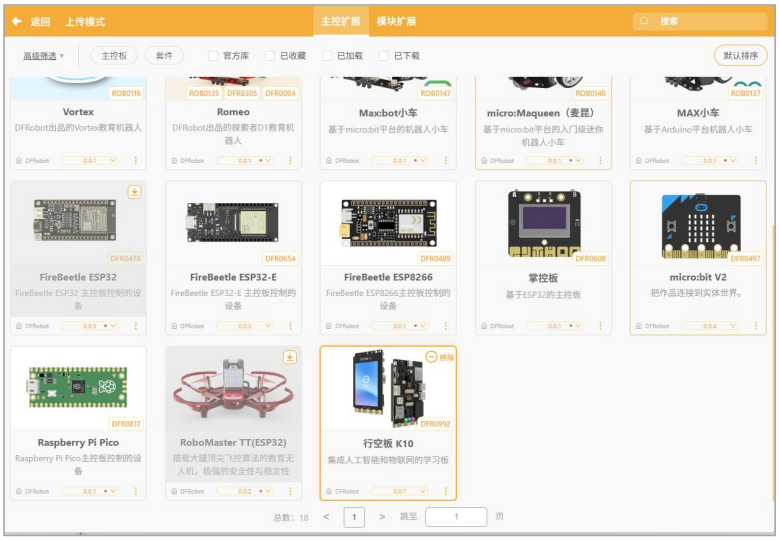
1. 硬件连接

将人体热释电红外传感器连接到行空板 K10 的 **P0** 端口，火焰传感器连接到行空板 K10 的 **P1** 端口。



2. 软件准备

1. 打开 **Mind+** 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**



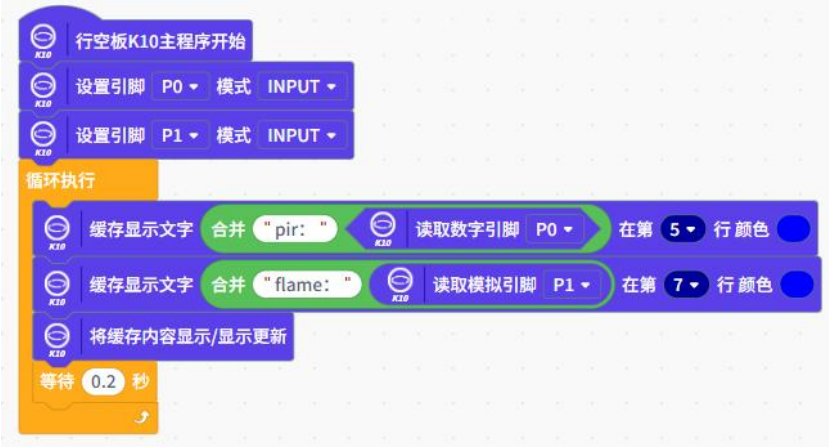
3. 程序编写

首先，需要使用"设置引脚 模式"指令，分别将 P0、P1 引脚设置为"INPUT"输入模式。

使用"读取数字引脚 P0"指令获取人体热释电红外传感器检测到的数值，然后用"合并 "指令在前面加上标签"pir："，拼成完整的显示文本，再通过"缓存显示文字"指令将其追加到缓存中。

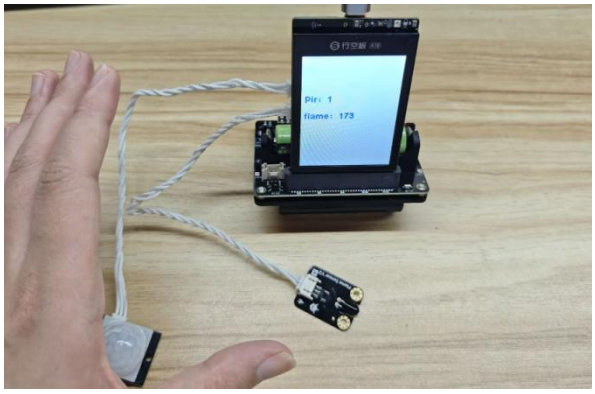
火焰传感器的显示也是一样的，使用"读取模拟引脚 P1"指令获取检测到火焰模拟值，用"合并 "指令加上标签"flame："，再用"缓存显示文字"追加到缓存。

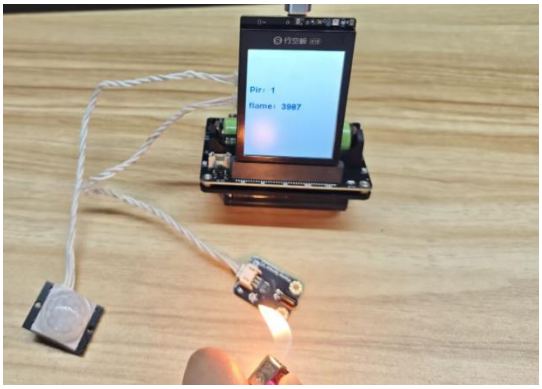
最后，使用“将缓存内容显示/显示更新”指令，一次性将所有内容显示在行空板 K10 的屏幕上。



4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10，观察屏幕上的数值变化。

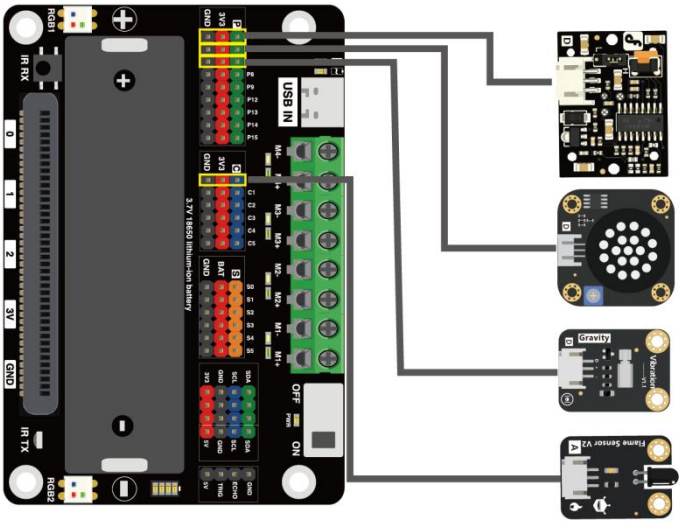




任务二：加入喇叭和振动模块，实现三防报警功能

1. 硬件连接

任务二加入了带功放喇叭模块和微型振动模块，需要使用**行空板多功能扩展板** DFR1216 来扩展接口。将人体热释电红外传感器连接到扩展板的 P0 端口，带功放喇叭模块连接到 P1 端口，微型振动模块连接到 P2 端口，火焰传感器连接到 C0 端口。



2. 软件准备

在任务一的基础上：

1. 在"模块扩展"中搜索 DFR1216，下载并添加
2. 在"模块扩展"中搜索 FIT0449，下载并添加

3. 程序编写

新建“变量 pir”和“变量 flame”，并初始化两个变量的值为 0。

将获取到的人体检测结果“读取模拟引脚 P0”指令赋值给“变量 pir”，将“读取引脚 C0 的模拟值”赋值给“变量 flame”。

接下来，使用“如果……那么执行……否则”指令对检测结果进行判断，当检测到有人（pir=1）或火焰值超过阈值（flame > 1500）时，使用“引脚 P1 播放音乐 重复”指令，控制喇叭播放报警音，使用“设置数字引脚 P2 输出高电平”指令，控制振动模块启动。

否则，条件为假时，说明没有检测到人体或者火焰，使用“引脚 P1 停止后台播放”指令，控制喇叭停止播放警报音；使用“设置数字引脚 P2 输出低电平”指令，控制振动模块停止振动。

最后，使用“缓存显示文字 ”指令与“将缓存内容显示/显示更新”指令，在行空板 K10 屏幕上，显示警报与安全的信息。完整程序如下：

注意：火焰传感器检测到火焰时输出值升高（接近 4095），无火焰时输出值较低（接近 0）。阈值 1500 可根据实际环境调整，数值越高表示火焰越近。

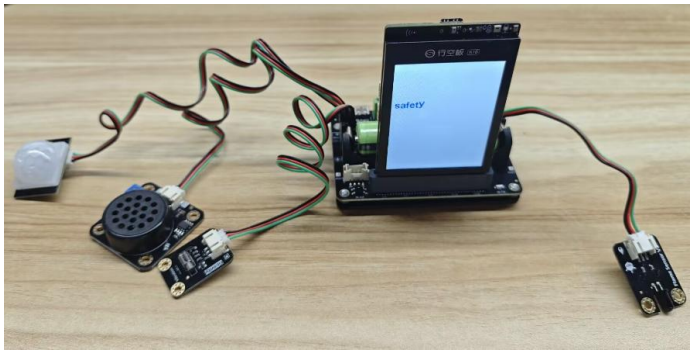
4. 程序运行

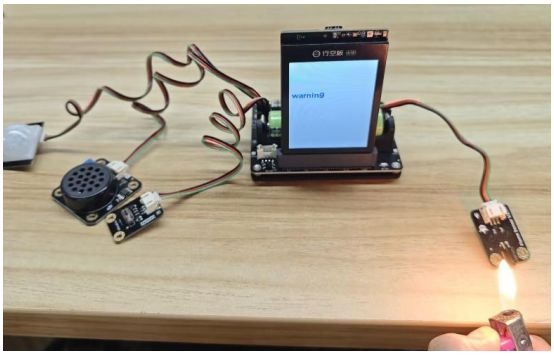
点击右上角"上传"按钮，将程序上传到行空板 K10。

人体入侵测试：用手在 PIR 传感器前晃动，观察屏幕显示"warning"，振动模块启动，喇叭发声

火焰测试：用打火机（不点火）靠近火焰传感器，观察是否触发报警

当检测不到人和火焰时，屏幕显示"safety"，所有报警停止





知识园地

人体热释电红外传感器（SEN0018）

人体热释电红外传感器（PIR，Passive Infrared）是一种**数字输入**传感器，用来检测人体发出的红外辐射。它的核心是一块热释电晶体——当人体移动时，晶体的温度变化会产生微弱的电荷信号，经过内部电路放大和比较后输出高电平（1）。

传感器前方覆盖着菲涅尔透镜（Fresnel Lens），它像一个"放大镜"，将人体发出的红外线聚焦到晶片上，同时将检测区域分成明暗相间的多个扇形区。人一旦走动，红外信号就会交替变化，传感器立刻就能"感知"到。这就是为什么静止不动时传感器可能检测不到，而挥手或走路时能准确触发。

火焰传感器（DFR0076）

火焰传感器是一种**模拟输入**传感器，用于检测火焰燃烧时产生的红外光（波长约760nm~1100nm）。它的核心是一个对红外光敏感的光电二极管——火焰越强，接收到的红外光越多，输出的模拟电压就越低（接近 0V）；无火焰时输出高电压（接近 3.3V）。

这就是程序中用 `flame > 1500` 来判断火焰的原因：数值越低表示火焰越近、越强。火焰传感器对日常灯光不敏感，但对打火机火焰、蜡烛火焰等红外成分丰富的光源反应灵敏。

带功放喇叭模块（FIT0449）

带功放喇叭模块是一种**模拟输出**执行器，内置功放芯片（放大器），可以直接播放声音。添加 FIT0449 库后，会提供"播放音符"指令——该指令将底层的 PWM 信号自动处理成中音和高音音符，编程时只需直接选择对应的音符即可，无需手动计算频率或占空比。

该库还提供了"播放音乐"指令，将 PWM 处理成了可直接选择的音乐。在本项目中我们直接选择一段连续的警报音乐，搭配振动模块实现"声+震"双重提醒。

微型振动模块（DFR0440）

微型振动模块是一种**数字输出**执行器，内部包含一个微型直流电机和偏心轮。当主控板输出高电平时，电机带动偏心轮高速旋转，产生离心力，使整个模块剧烈振动——就像手机调成振动模式时的感觉。

高电平（1）：电机通电，模块振动

低电平（0）：电机断电，振动停止

振动模块特别适合做无声报警——例如深夜检测到入侵时，用振动提醒用户而不惊动入侵者。

拓展挑战

1. **调整阈值**：改变判断条件中的阈值，观察效果变化。
2. **多条件组合**：组合多个传感器的数据进行判断，提高准确性。

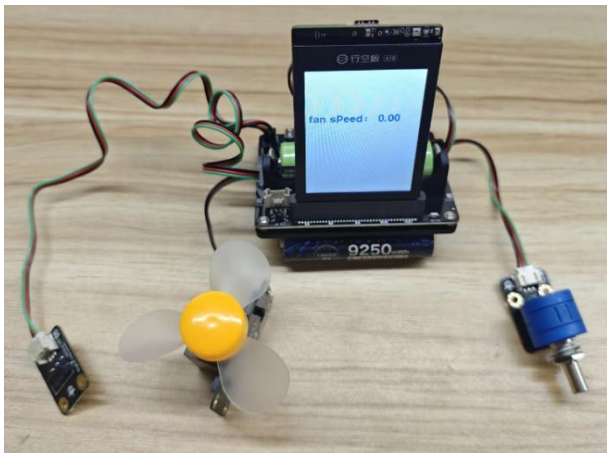
第 6 课 随动智控风扇

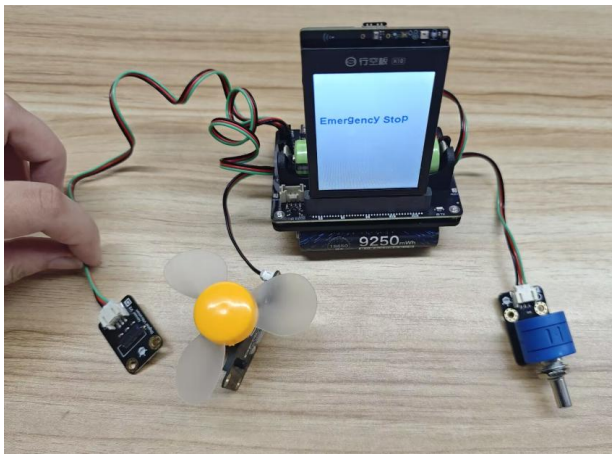
夏天坐在书桌前，风扇呼呼地吹，但风不是太大就是太小。想要调个合适的风速，还得起身去按风扇的档位按钮。如果转一下旋钮风扇就跟着变快，拍一下桌子风扇就立刻停止，是不是方便多了？

随动智控风扇就是这样一个"听你指挥"的智能风扇。本项目将使用角度传感器检测旋转角度来控制风扇转速，同时用震动传感器实现紧急停止功能。

任务目标

设计并制作一个随动智控风扇：转动角度传感器调节风扇转速，震动传感器触发紧急停止。





学习目标

- 学会使用模拟角度传感器检测旋转角度

- 学会使用直流电机风扇模块
- 学会使用数字震动传感器检测震动信号
- 理解模拟输入控制数字输出的编程方法

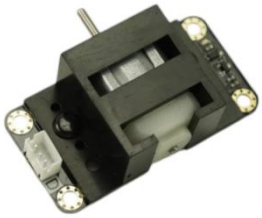
材料清单

硬件清单

行空板 K10(DFR0992) x 1

行空板多功能扩展板 (DFR1216) x 1

模拟角度传感器(DFR0058) x 1



130 直流电机风扇（DFR0411) x 1

数字震动传感器（DFR0027）x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

这个项目，主要是通过下面两个任务，逐步完成随动智控风扇的制作与调试。在任务一中，读取角度传感器和震动传感器的数值并通过屏幕观察。在任务二中，加入风扇模块，实现角度控制转速、震动急停的功能。

任务一：读取角度传感器和震动传感器的值

读取两个传感器输出的数值，观察旋转角度和震动对应的数值变化规律。

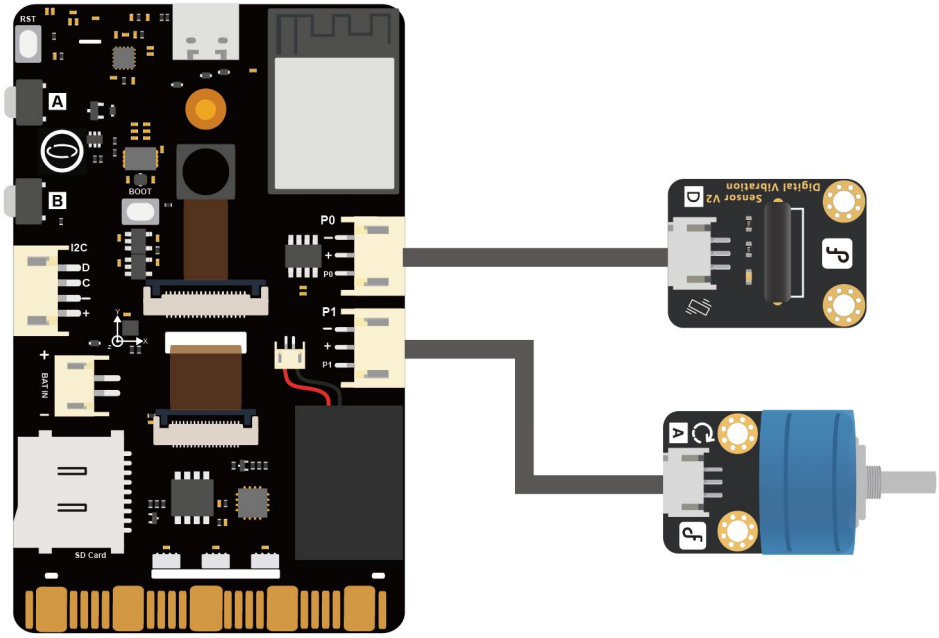
任务二：角度控制风扇转速

在任务一的基础上，加入风扇模块，实现角度传感器控制风扇转速、震动传感器急停的功能。

任务一：读取角度传感器和震动传感器的值

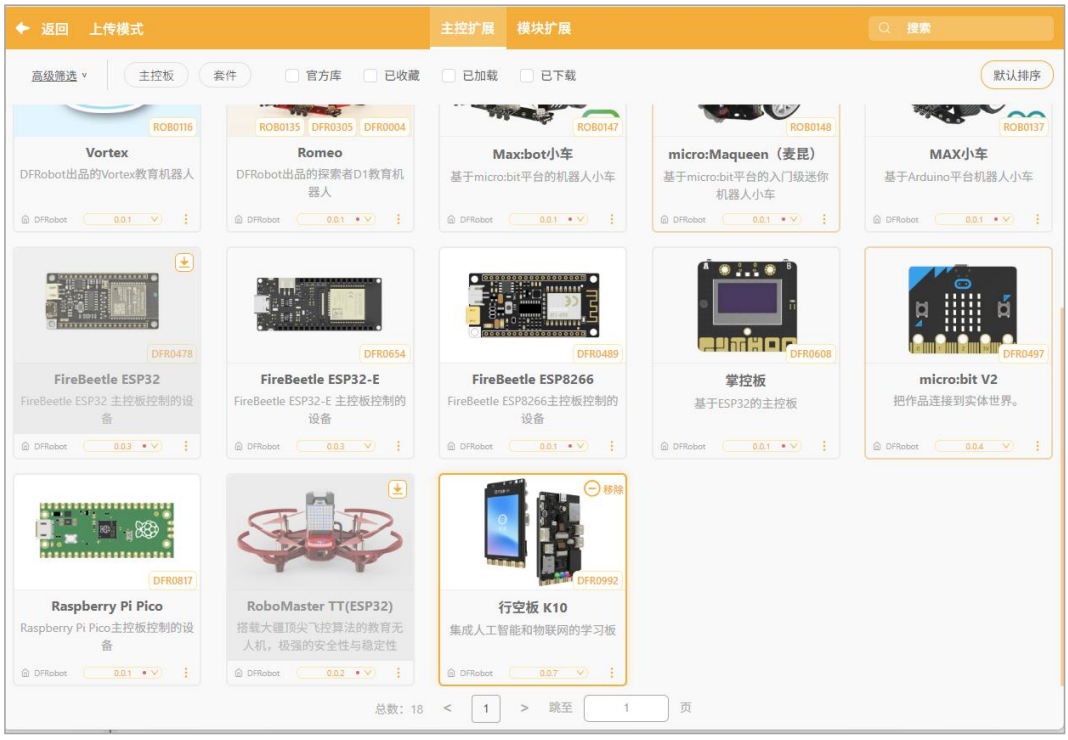
1. 硬件连接

将角度传感器连接到行空板 K10 的 **P0** 端口，震动传感器连接到行空板 K10 的 **P1** 端口。



2. 软件准备

1. 打开 **Mind+** 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**



3. 程序编写

首先，使用"设置引脚 模式"指令，分别将 P0、P1 引脚设置为"INPUT"输入模式。

使用"读取模拟引脚 P0"指令获取角度传感器的模拟值，用"合并"指令加上标签"angle：", 再用"缓存显示文字"指令将其追加到缓存。

使用"读取数字引脚 P1"指令获取震动传感器检测到的数值，然后用"合并"指令在数值前面加上标签"shake：", 拼成完整的显示文本，再通过"缓存显示文字"指令将其追加到缓存中。

最后，使用"将缓存内容显示/显示更新"指令，一次性将所有内容显示在行空板 K10 的屏幕上。

4. 程序运行

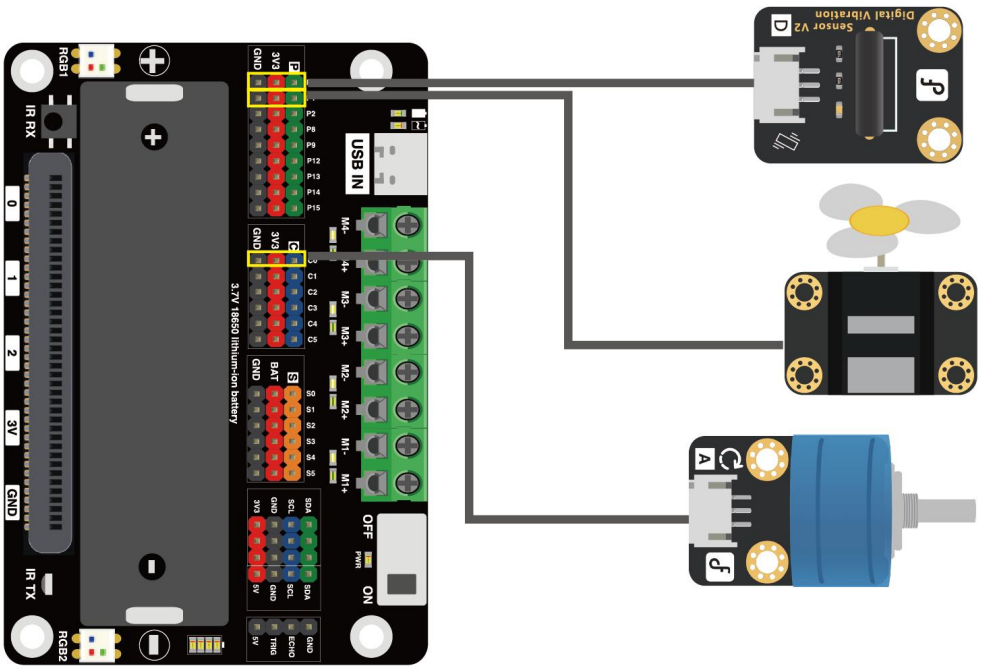
点击右上角"上传"按钮，将程序上传到行空板 K10。观察屏幕上两个传感器的数值。

转动角度传感器的旋钮，观察角度值在 0~4095 之间变化。敲击桌面或触动震动传感器，观察震动值从 1 变为 0。

任务二：角度控制风扇转速

1. 硬件连接

任务二加入了电机风扇模块，需要使用**行空板多功能扩展板 DFR1216** 来扩展接口。将角度传感器连接到扩展板的 C0 端口，震动传感器连接到 P0 端口，电机风扇连接到 P1 端口。



2. 软件准备

在任务一的基础上，在"模块扩展"中搜索 DFR1216，下载并添加。

3. 程序编写

创建三个变量：fan_speed（转速值 0~1023）、 shake（振动检测）、angle（角度检测）。

使用"读取引脚 C0 的模拟值"指令获取角度传感器的模拟值，存入变量 angle；使用"读取数字引脚 P0"指令获取震动传感器的数字值，存入变量 shake。

接下来使用"如果...那么...否则"指令进行判断：当 shake = 0 时（检测到震动），执行紧急停止。使用"设置模拟引脚 P1 输出（PWM）"指令让风扇紧急停止转动。

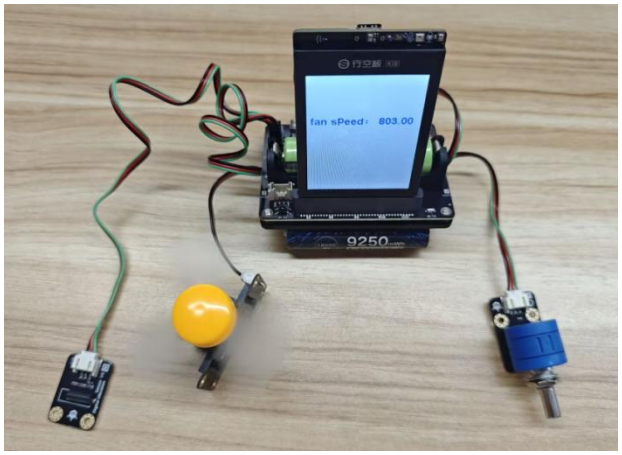
否则（未检测到震动），使用“映射 angle 从(0,4095)到(0,1023)”指令，将 angle 从 0~4095 映射到 0~1023，对应风扇的 PWM 转速值。

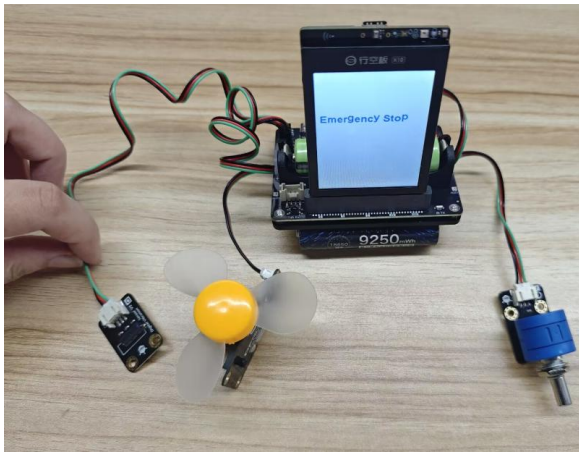
使用“设置模拟引脚 P1 输出”指令，控制风扇的转速为“变量 fan_speed”。

最后，我们还可以通过“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，在行空板 K10 屏幕上显示“急停”与风扇转速的文字做提示。

4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。缓慢旋转角度传感器，观察风扇转速是否随之变化。用力敲击桌面（或触碰震动传感器），观察风扇是否立即停止。





知识园地

角度传感器——像音量旋钮一样

角度传感器（也叫电位器）内部是一个可变电阻器。旋转旋钮时，电阻值会平滑变化，主控板读取到的模拟电压也随之改变。0°时输出 0V，旋转到最大角度时输出 3.3V。

你可以把它想象成收音机的音量旋钮——旋到不同位置，音量（风扇转速）就不同。正是这种"连续变化"的特性，让它非常适合做调速控制。

震动传感器——灵敏的"触觉"

数字震动传感器内部有一个弹簧和金属触片。当传感器静止时，电路导通，输出高电平（1）；当受到震动时，弹簧晃动接触金属片断开，电路瞬间断开，输出低电平（0）。

它就像一个灵敏的"触觉器官"，能感知到桌面敲击、物体碰撞等震动事件。

模拟输出与 PWM 调速

行空板 K10 的模拟输出实际上是 PWM（脉冲宽度调制）信号。它通过快速开关电源，让"开"的时间比例变化，从而控制电机转速：

占空比 0%（一直关）→ 电机停止

占空比 50%（一半时间开）→ 电机半速

占空比 100%（一直开）→ 电机全速

数值映射——让不同范围"对齐"

角度传感器输出 0~4095 的模拟值，而 PWM 调速只接受 0~1023。两个范围不同，直接使用会出错——旋钮转到一半（约 2048），风扇不会转一半，而是会"看不懂"这个数。

解决办法就是**映射**：将 4095÷4≈1023，使 0~4095 对应到 0~1023。

可以把映射想象成"翻译官"：角度传感器说"2048"，翻译官÷4 后告诉风扇"512"，风扇就知

道转一半速度。整条链路就是：**旋动角度传感器 → 读取模拟值 0~4095 → 除以 4 映射到 0~1023 → PWM 输出控制风扇转速**。

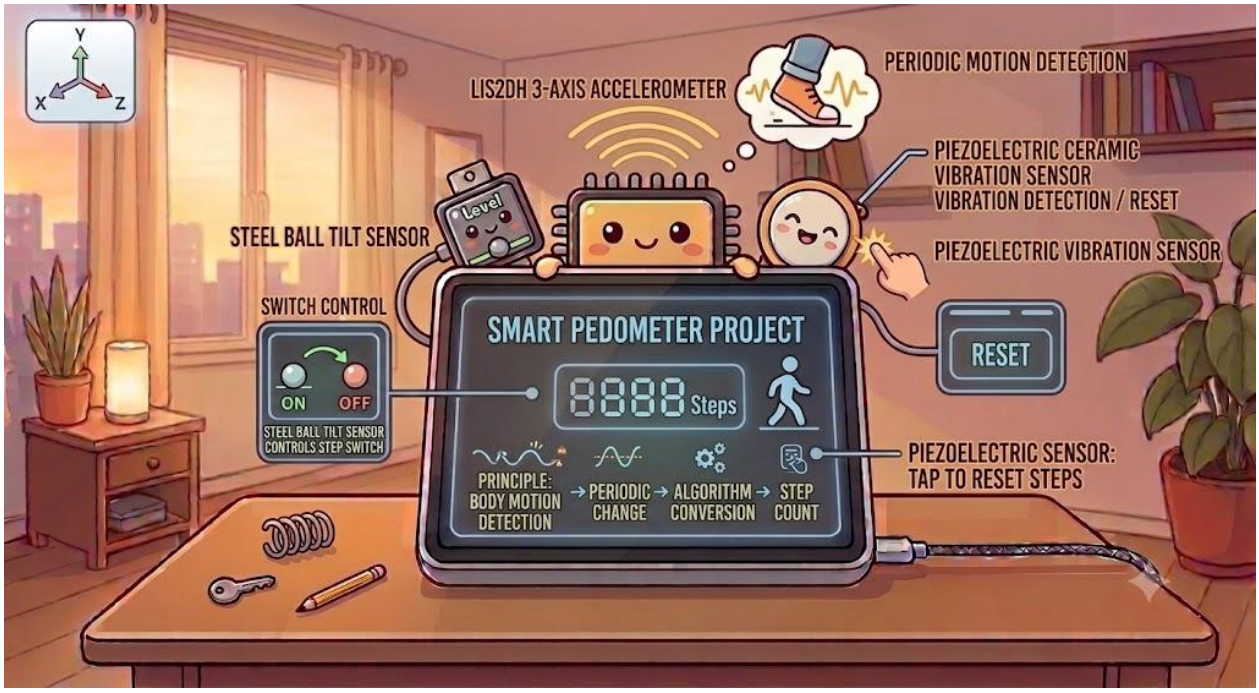
拓展挑战

1. **震唤醒**：风扇停止后，轻触震动传感器重新启动风扇，恢复上次的转速。
2. **最低转速保护**：设置一个最低转速阈值，避免角度过小时风扇发出嗡嗡声但不转动。

第 7 课 智能计步器

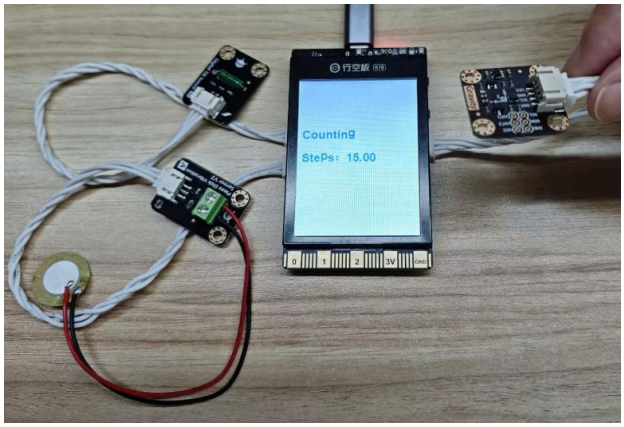
你有没有想过，手机和手环是怎么知道你走了多少步的？答案就藏在里面一个小小的加速度计里——走路时身体会上下起伏，加速度计检测到这种周期性变化，再通过算法转换成步数。

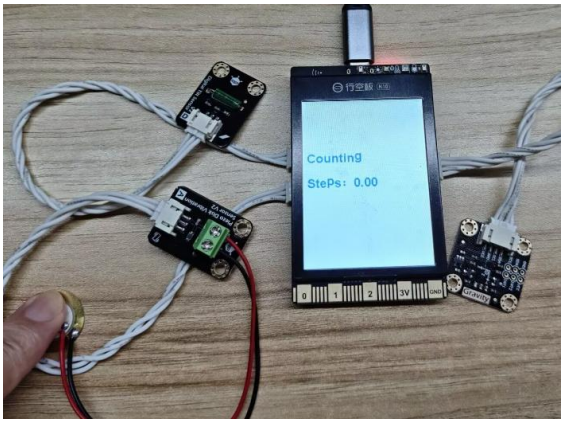
智能计步器就是一个能"感知脚步"的神奇装置。本项目将使用 LIS2DH 三轴加速度计检测步行时的加速度变化来实现计步，钢球倾角传感器控制计步开关，压电陶瓷振动传感器敲击复位步数。



任务目标

设计并制作一个智能计步器：利用三轴加速度计检测走路时的加速度变化，实现计步功能，钢球倾角传感器控制计步的开启和暂停，压电陶瓷振动传感器敲击复位步数。





学习目标

- 学会使用 I2C 三轴加速度计读取 X/Y/Z 轴加速度数据
- 掌握加速度计计步的基本原理和编程方法
- 学会使用数字倾角传感器作为开关控制程序状态
- 学会使用模拟振动传感器检测敲击信号

材料清单

硬件清单

行空板 K10(DFR0992) x 1

行空板多功能扩展板 (DFR1216) x 1

三轴加速度计(SEN0224) x 1

数字钢球倾角传感器（DFR0028) x 1

模拟压电陶瓷振动传感器（DFR0052） x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

本项目主要通过下面三个任务，逐步完成智能计步器的制作与调试。在任务一中，学会读取加速度计的三轴数据并通过屏幕观察。在任务二中，利用 Z 轴数值实现基础计步功能。在任务三

中，加入倾角传感器和振动传感器，实现计步开关和步数清零。

任务一：读取加速度计三轴数据

读取 LIS2DH 三轴加速度计的 X、Y、Z 值，在屏幕显示，走路时观察数值的变化规律。

任务二：实现基础计步

利用加速度计 Z 轴数据检测走路时的上下振动，通过阈值判断实现计步，步数显示在屏幕上。

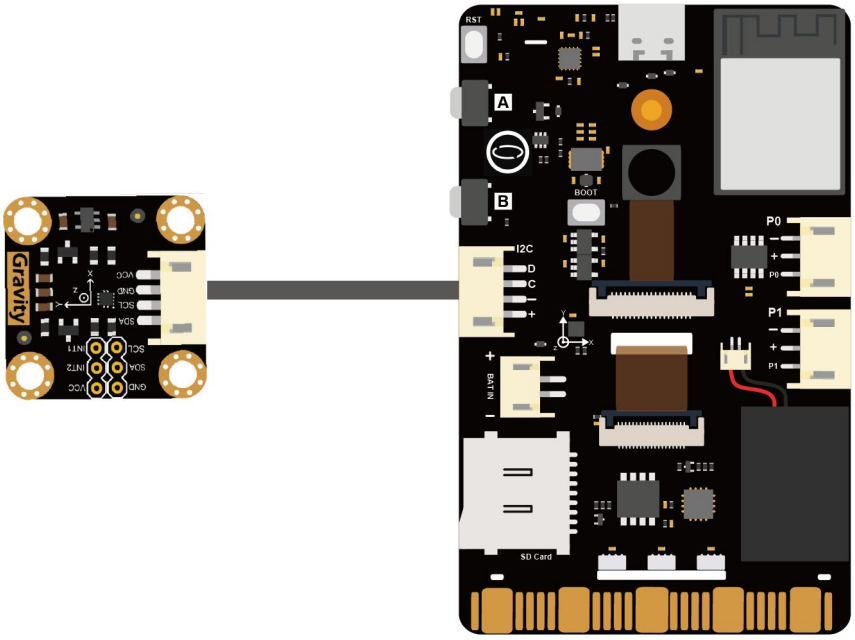
任务三：加入计步开关和步数清零

在计步基础上加入钢球倾角传感器控制计步启停，压电陶瓷振动传感器敲击复位步数。

任务一：读取加速度计三轴数据

1. 硬件连接

将三轴加速度计连接到行空板 K10 的 **I2C 接口**。

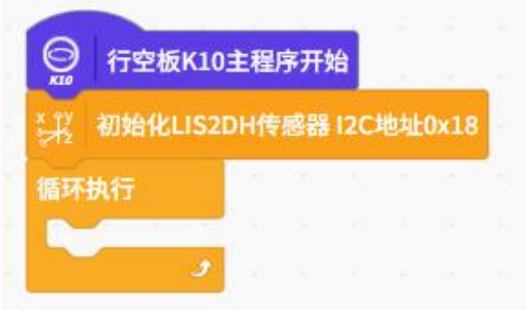


2. 软件准备

1. 打开 **Mind+** 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**
4. 在"模块扩展"中搜索 **SEN0224**，下载并添加

3. 程序编写

首先，在初始化模块中添加"初始化 LIS2DH 传感器 I2C 地址 0x18"指令，完成加速度计的初始化。

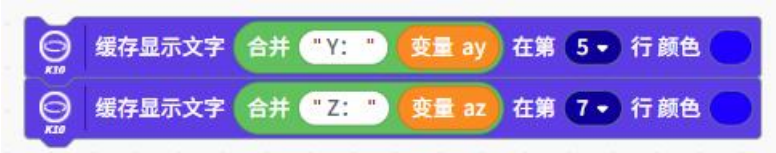


新建“变量 ax”、“变量 ay”、“变量 az”，并初始化三个变量的值为 0。

在“循环执行 ”中，使用“读取 LIS2DH 传感器 x 轴加速度值（mg）”获取检测到的加速度值，将获取的值赋值给对应的变量。

接着使用“缓存显示文字”指令和“合并 ”指令，在屏幕上的第 3 行，显示合并后的标签"X: "和检测到的 x 轴加速度值（变量 ax）。

同样的方式在第 5 行显示“Y”和 ay，在第 7 行显示“Z”和 az。



最后，使用“将缓存内容显示/显示更新”指令，将缓存中的数据显示出后，并加上“等待 0.2 秒”，让屏幕稳定显示。完整程序如下：

4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。观察屏幕上三轴加速度计的数值。

- 将装置水平静置在桌面上，观察 X、Y、Z 的数值——Z 轴约等于重力加速度值。

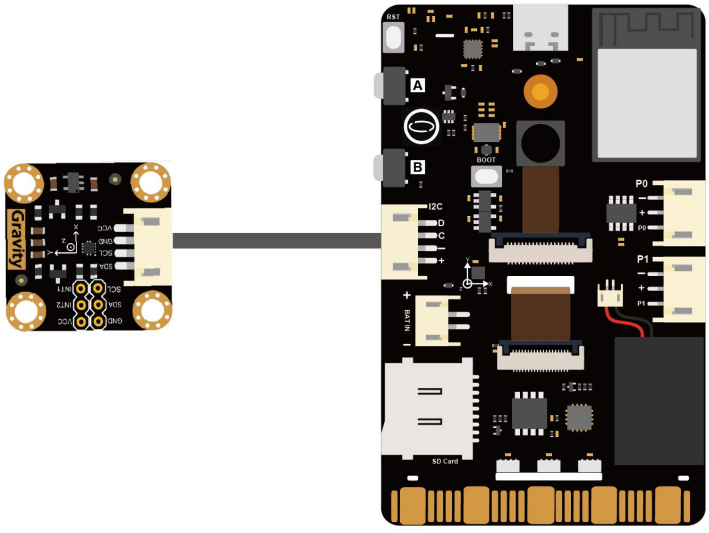
- 拿起装置在手中，快速上下抖动，观察 Z 轴数值如何跳动。
- 正常走路时手持或佩戴装置，观察 Z 轴数值的周期性变化规律。
- 记下静止时的 Z 轴基线和走路时的峰值，为任务二设置阈值做准备。

任务二：实现基础计步

在任务一的基础上，加入计步算法，利用 Z 轴数值的上升沿检测实现基础计步功能。

1. 硬件连接

保持任务一的硬件连接不变。



2. 程序编写

计步器的核心原理：走路时身体会上下起伏，加速度计的三轴数值都会发生变化。使用**模长公式**即 ` $\sqrt{(X^2+Y^2+Z^2)}$ `将三轴数据合并为一个总加速度值。无论设备如何摆放，走路时的总加速度都会呈现周期性波峰，检测到波峰即为一步。

创建三个变量：**mag** 存储总加速度模长，**steps** 存储步数，**last_mag** 存储上一次的总加速

度值用于上升沿检测。

接下来，计算总加速度模长：

在运算符中，使用“ * ”指令，先分别计算三个轴数值的平方；

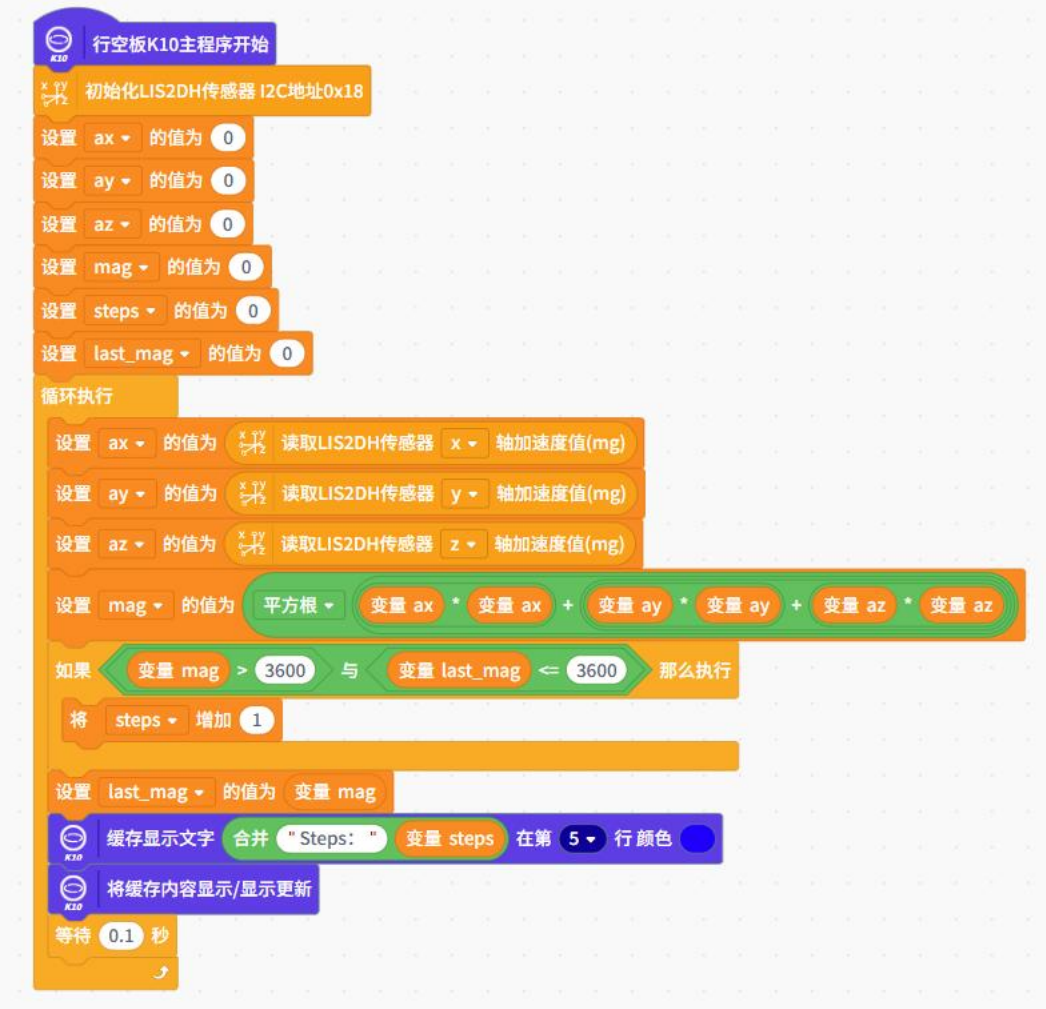
使用“ + ”指令，对三个轴的平方进行求和；

最后使用“平方根 ”指令，对应求和数据进行开平方，并将结果存入变量 mag。

当 mag 大于设定阈值（例如，3600）且 last_mag 小于等于阈值时（3600），使用“将 steps 增加 ”指令，将步数加 1。

然后使用“设置 last_mag 的值为 ”指令，更新变量 last_mag 的值。使用“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，进行显示步数。

完整程序如下：

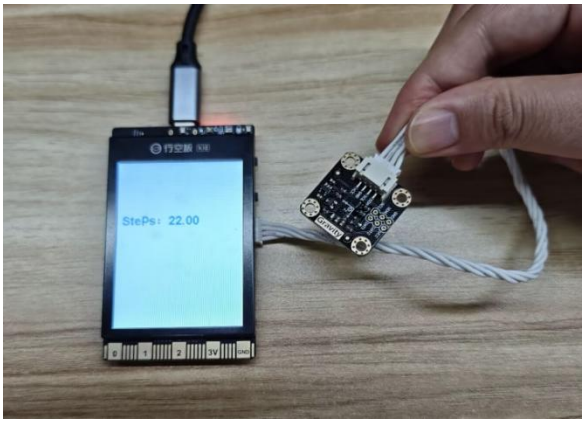


计步原理说明：走路时身体上下起伏，总加速度模长 ` $\sqrt{(X^2+Y^2+Z^2)}$ ` 会周期性变化。静止时模长约等于重力加速度（约 3600），走路时模长在波峰处超过静止值。程序通过检测上升沿来计步，避免重复计数。阈值需根据任务一中观察到的静止基线和走路峰值调整。

3. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。

- 正常走路手持或佩戴装置，观察屏幕步数是否增加。
- 尝试不同速度走路，观察步数计数效果。
- 如果步数计数不准确（多计或少计），调整程序中的阈值后重新上传测试。

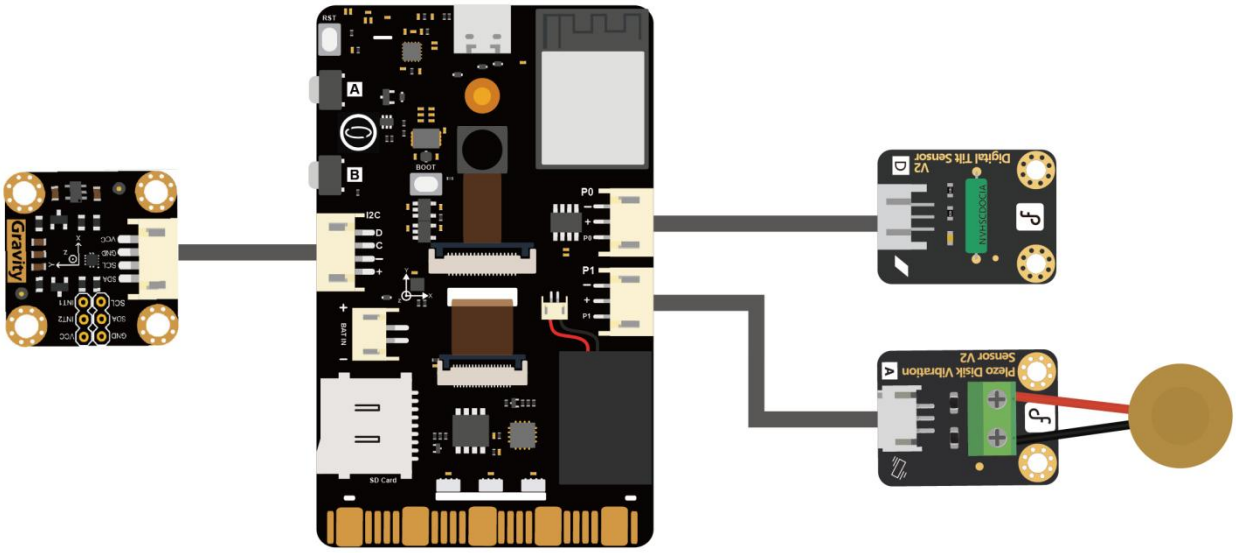


任务三：加入计步开关和步数清零

在任务二的基础上，加入钢球倾角传感器控制计步启停，压电陶瓷振动传感器敲击复位步数。

1. 硬件连接

任务三加入了倾角传感器和振动传感器，需要使用**行空板多功能扩展板** DFR1216 来扩展接口。将钢球倾角传感器连接到扩展板的 P0 端口，压电陶瓷振动传感器连接到 P1 端口，三轴加速度计连接到扩展板的 I2C **接口**。



2. 软件准备

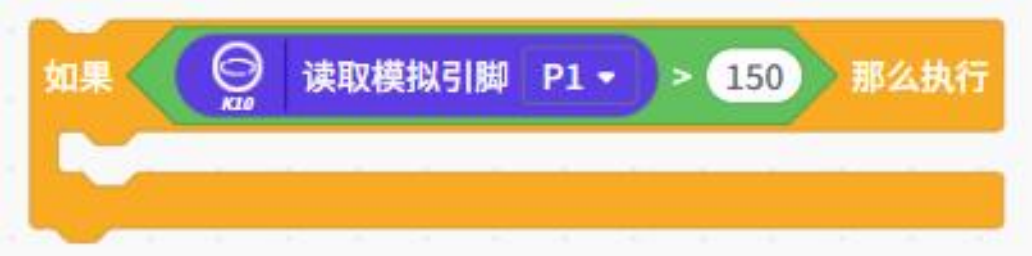
在任务二的基础上，在"模块扩展"中搜索 DFR1216，下载并添加。

3. 程序编写

在任务二的计步逻辑基础上，增加两个控制功能：步数清零与计步开关

1. **步数清零**

使用“如果……那么执行”指令，判断“读取模拟引脚 P1 > 150”，是否检测到有敲击。



条件为真时，说明检测到敲击，使用“设置 steps 的值为 ”指令，设置变量 steps 的值为 0，将步数归零。

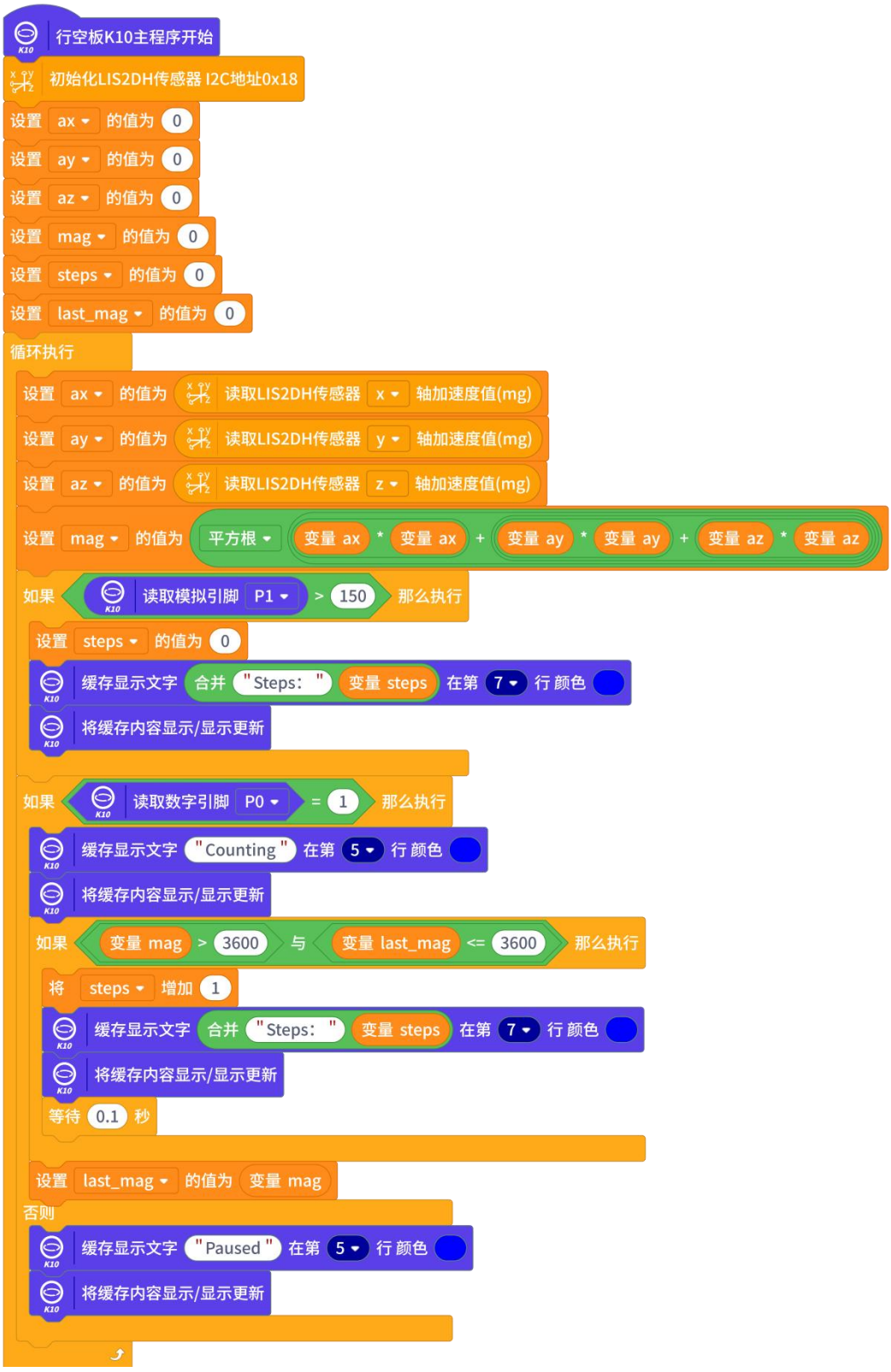
2. **计步开关**

使用“如果……那么执行……否则”指令，判断“读取数字引脚 P0 = 1”，是否检测到倾角传感器有倾斜。

条件为真时，开始计步检测（任务二中的计步程序），并使用“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，显示当前模式为“Counting”。

否则，暂停计步，并使用“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，显示当前模式为“Paused”。

完整程序如下：



4. 程序运行

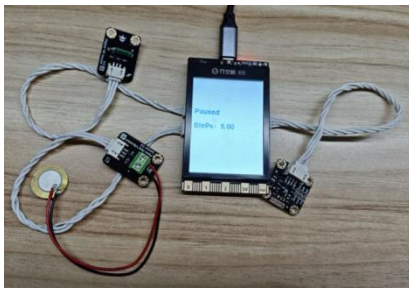
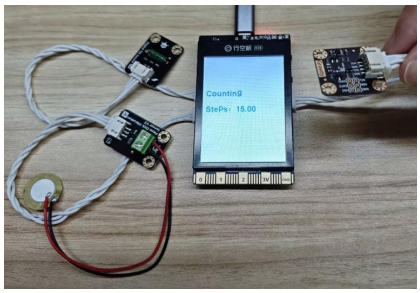
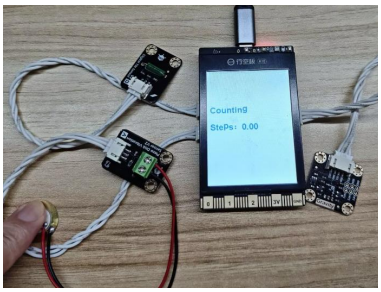
点击右上角"上传"按钮，将程序上传到行空板 K10。

- **开始计步**：将装置倾斜佩戴或手持，屏幕显示"Counting..."，正常走路观察步数是否增加。
- **暂停计步**：将装置水平放置在桌面上（钢球倾角传感器不倾斜），屏幕显示"Paused"，

步数停止增加。

- **复位步数**：敲击或弹压压电陶瓷振动传感器，屏幕显示"Reset!"，步数归零。

如果步数计数不准确（多计或少计），调整程序中的阈值（2200）后重新上传测试。

知识园地

加速度计是如何计步的？

LIS2DH 三轴加速度计内部有一个微小的 MEMS（微机电系统）结构，可以检测 X、Y、Z 三个方向的加速度变化。

走路时，我们的身体会随着步伐上下起伏——脚踩地时身体向上，脚离地时身体向下。这个上下运动的加速度变化主要反映在 Z 轴数据上：

- **静止时**：Z 轴数值基本稳定（约等于 1g 对应的数值）。
- **走路时**：Z 轴数值周期性波动，每一步产生一个波峰和一个波谷。

计步器的核心就是把这种周期性波峰"数"出来。程序通过**上升沿检测**——当数值从低于阈值变为高于阈值时计为一步，并配合消抖避免重复计数。

三种传感器，三种角色

本项目用到了三种不同类型的传感器，在计步器中各司其职：

LIS2DH 三轴加速度计（I2C 通信）是核心传感器，通过 SDA/SCL 两根数据线传输 X、Y、Z 三轴数值。功能最强大，可以读取多维数据，精度高。

钢球倾角传感器（数字信号）内部有一颗小钢珠。水平时钢珠停留在底部，电路断开（输出 0）；倾斜时钢珠滚向一侧接触触片，电路导通（输出 1）。在计步器中用作开关——水平放置时暂停计步，倾斜时开始计步。

压电陶瓷振动传感器（模拟信号）利用压电效应：陶瓷片受到挤压或振动时表面产生微弱的电压，振动越强电压越高。在计步器中用来检测敲击，实现复位操作。

从 I2C 到数字到模拟，三种信号类型恰好代表了传感器与主控通信的三种主流方式。

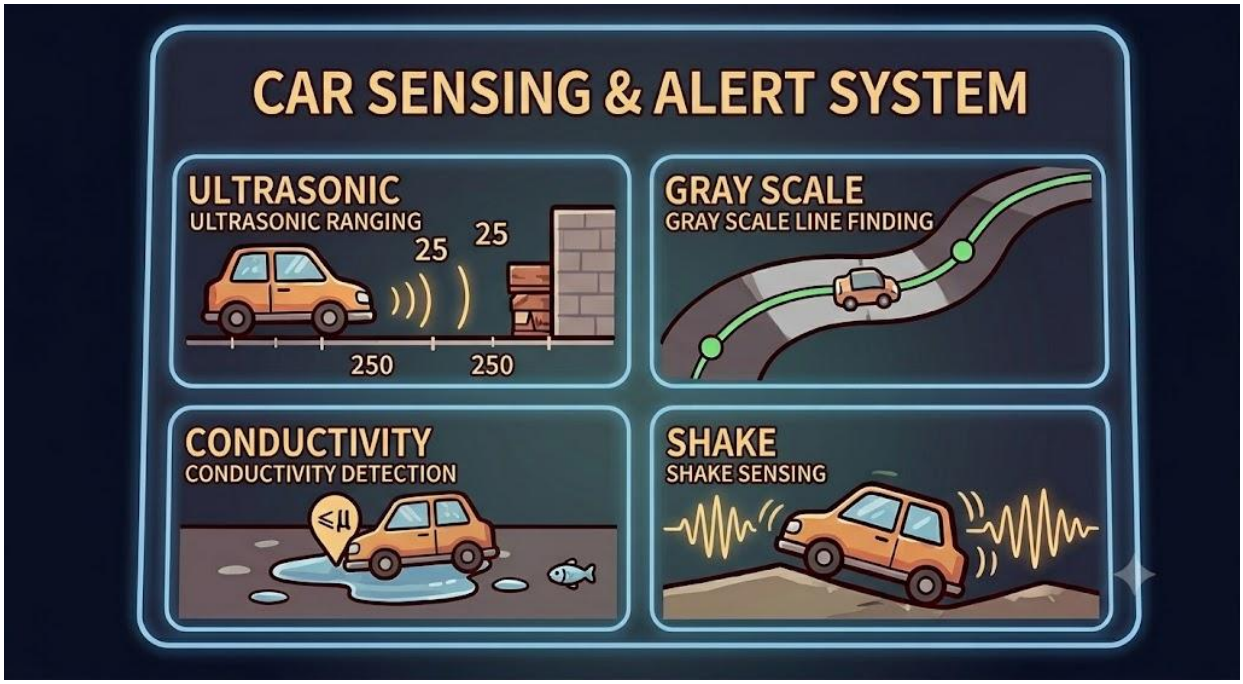
拓展挑战

1. **调整灵敏度**：在任务一中记录走路和跑步时 Z 轴的不同峰值，分别设置"走路模式"和"跑步模式"两个阈值。
2. **双轴计步**：除了 Z 轴，尝试用 X 轴或 Y 轴的组合来判断步数，看看能否减少误计。
3. **计步器倒计时**：设定目标步数（如 100 步），达到目标后屏幕显示"Goal!"鼓励文字。

第 8 课 汽车路况感知系统

你有没有想过，一辆智能汽车是如何感知周围环境的？倒车时雷达自动报警、偏离车道时发出警示、遇到积水路面自动提示、发生碰撞时触发紧急响应——这一切都靠车上的各种传感器协同工作。

汽车路况感知系统就是用超声波测距、灰度寻线、导电检测和晃动感知四种传感器，模拟一辆智能汽车的环境感知功能。每个传感器对应汽车的一个感知子系统，协同判断车辆状态。



任务目标

设计并制作一个汽车路况感知系统：超声波测距模拟倒车雷达，灰度传感器模拟车道检测，电导开关模拟积水路面检测，晃动传感器模拟碰撞检测，四种数据融合判断并发出声光警报。

学习目标

- 学会使用模拟量超声波测距传感器检测距离
- 学会使用模拟灰度传感器识别黑白颜色
- 学会使用电导开关模块检测导电
- 学会使用数字晃动传感器检测碰撞
- 掌握多传感器数据融合与联动判断的编程方法

材料清单

硬件清单

行空板 K10(DFR0992) x 1

行空板多功能扩展板 (DFR1216) x 1

模拟量超声波测距传感器 (SEN0307) x 1

模拟灰度传感器（DFR0022) x 1

电导开关模块（SEN0223） x 1

数字晃动传感器（SEN0289） x 1

带功放喇叭模块（FIT0449） x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

本项目主要通过下面三个任务，逐步完成汽车路况感知系统的制作与调试。在任务一中，制

作一个倒车雷达——超声波测距驱动喇叭报警。在任务二中，加入灰度传感器，增加车道偏离预警功能。在任务三中，加入电导开关和晃动传感器，实现完整的汽车感知融合系统。

任务一：倒车雷达

用超声波传感器检测距离，障碍物太近时喇叭发出急促警报声。

任务二：车道偏离预警

在倒车雷达基础上加入灰度传感器，检测到黑色（压线）时喇叭发出长响警报。

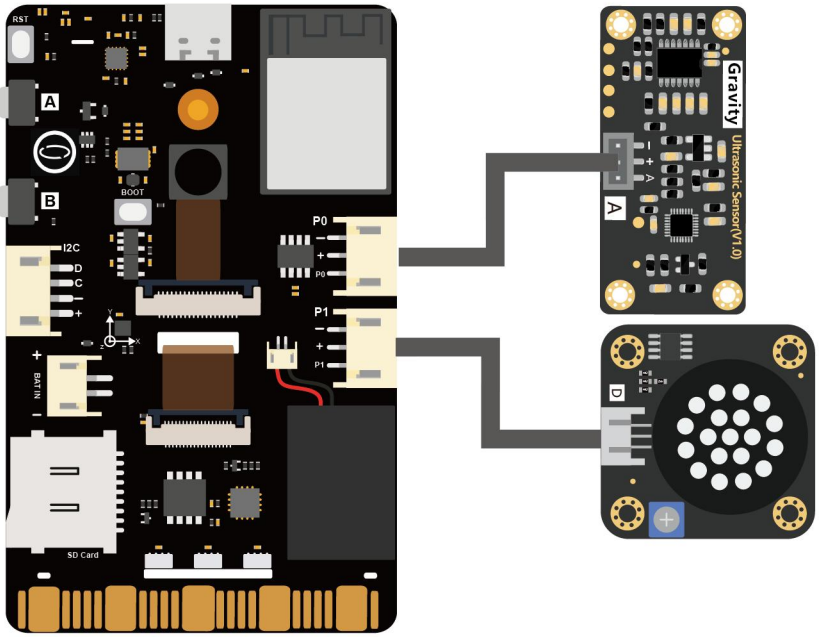
任务三：全车感知融合系统

加入电导开关和晃动传感器，模拟涉水检测和碰撞警报，四种传感器数据融合判断。

任务一：倒车雷达

1. 硬件连接

将超声波测距传感器连接到行空板 K10 的 P0 端口，喇叭连接到 P1 端口。



2. 软件准备

1. 打开 Mind+ 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**
4. 在"模块扩展"中搜索 SEN0307，下载并添加超声波测距传感器扩展库
5. 在"模块扩展"中搜索 FIT0449，下载并添加带功放喇叭模块扩展库

3. 程序编写

首先使用"设置引脚 模式"指令，将 P0 设置为"INPUT"输入模式，P1 设置为"OUTPUT"输出模式。

创建"变量 dist"，用来存放超声波传感器检测到距离数据，并在主程序开始下初始化该变量的值为 0。

在“重复执行”中，使用模拟超声波扩展库中的指令“读取 P0 模拟量超声波（cm）”，获取检测到的距离数据，并将数据赋值给“变量 dist”。

使用“如果……那么执行……否则”指令，对变量 dist 进行判断。当数值小于 30 时，认为有障碍物，使用“引脚 P1 播放音乐 重复在后台播放一次”指令，控制喇叭发出“DADADADUM”的声音。

否则，使用“引脚 P1 停止后台播放”指令，控制喇叭停止播放。

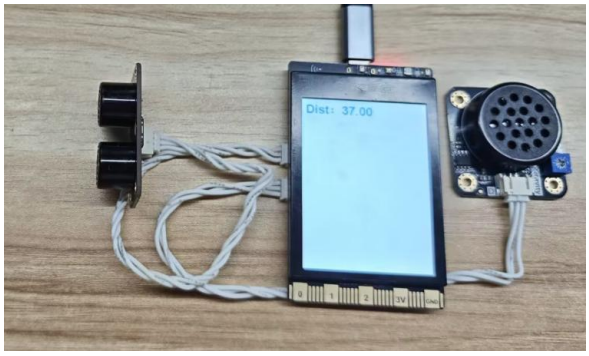
最后，使用“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，将超声波传感器检测到的距离显示在行空板 K10 屏幕上。

4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。

将手放在超声波传感器前方慢慢靠近，距离数值越小喇叭响得越急促。

记下不同距离对应的数值范围，为后续任务设置更精准的阈值。

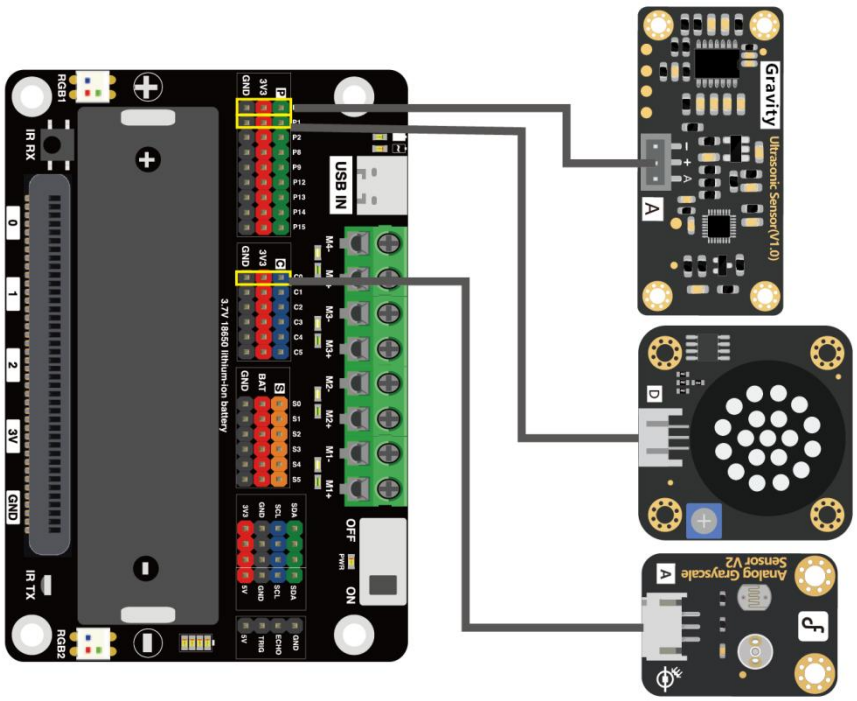


任务二：车道偏离预警

在任务一的基础上，加入灰度传感器模拟车道偏离检测——当压到黑色车道线时喇叭发出长响警报。

1. 硬件连接

保持任务一的硬件连接不变，只是将超声波传感器和带功放喇叭模块，连接到扩展板上的 P0、P1 就，将灰度传感器连接到扩展板的 C0 端口。



2. 软件准备

在任务一的基础上，在"模块扩展"中搜索 DFR1216，下载并添加行空板多功能扩展板扩展库。

3. 程序编写

创建“变量 dist”、“变量 gray”，并初始化两个变量的值为 0。

在“重复执行”指令中，使用“读取 P0 模拟量超声波（cm）”指令，获取距离值并将该值赋值给变量 dist。使用“读取引脚 C0 的模拟值”指令，获取灰度传感器检测到的数据并赋值给变量 gray。

在实际情况中，倒车雷达优先级高于车道偏离——障碍物太近比压线更紧急，距离触发时不再检查灰度，两个条件同时满足时，只执行倒车雷达警报。

因此，使用“如果……那么执行……否则如果……那么……”指令，先判断离障碍物是否小于 30。条件为真，喇叭播放音乐“DADADADUM”。

接下来，再判断小车是否压线（变量 gray > 2500）。条件为真，喇叭播放音乐“PRELUDE”。

最后，使用“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，将超声波传感器检测到的距离，灰度传感器检测到的灰度数据，显示在行空板 K10 屏幕上。

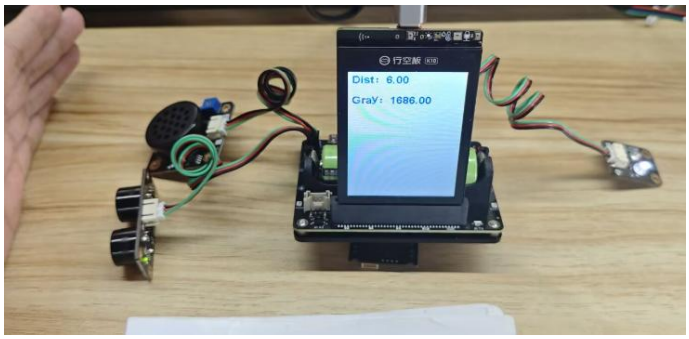
4. 程序运行

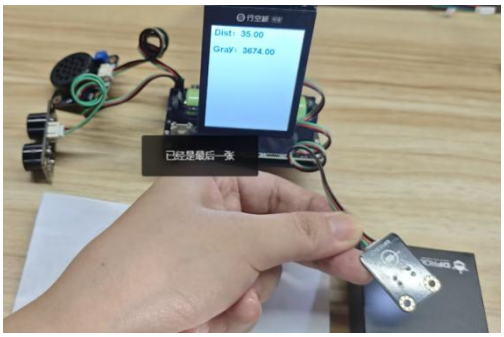
点击右上角"上传"按钮，将程序上传到行空板 K10。

倒车雷达：手靠近超声波传感器，喇叭急促响。

车道偏离：将灰度传感器对准黑色表面，喇叭长响一声。

尝试同时触发两个警报，观察喇叭的声音变化。



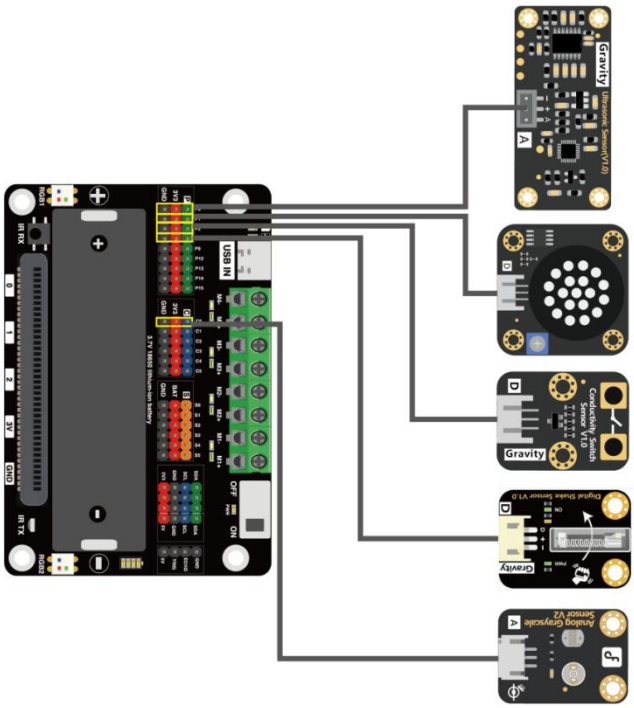


任务三：全车感知融合系统

在任务二的基础上，加入电导开关和晃动传感器，模拟涉水检测和碰撞警报，完成四种传感器的数据融合。

1. 硬件连接

任务三需要使用**行空板多功能扩展板 DFR1216** 来扩展接口。将超声波测距传感器连接到扩展板的 P0 端口，带功放喇叭模块连接到扩展板的 P1 口，灰度传感器连接到扩展板的 C0 端口，电导开关模块连接到 P2 端口，喇叭连接到 P3 端口，晃动传感器连接到 P8 端口。



3. 程序编写

四种传感器分别模拟汽车的四个感知子系统：

传感器	汽车子系统	触发条件	警报音
超声波	倒车雷达	距离 < 30	DADADADUM
灰度	车道偏离预警	灰度 < 2500	PRELUDE
电导	涉水检测	导电 = 1	ENTERTAINER
晃动	碰撞检测	晃动 = 0	ODE

新建“变量 shake”、“变量 conduct”，并初始化变量的值为 0。

在重复执行中，读取四个传感器的数值。

按优先级依次判断警报条件。优先级顺序：倒车雷达 > 碰撞 > 车道偏离 > 涉水。高优先级条件触发时不再检查低优先级条件。

完整程序如下：

- **传感器融合的思路**：四种传感器各自检测环境的不同维度。单独看它们只是四个独立的数据，但放在"汽车感知系统"的场景中，它们共同构建了一幅完整的车辆环境图。这就是多传感器融合（Multi-Sensor Fusion）的核心理念——用多个传感器的互补信息，做出更准确的综合判断。

4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。

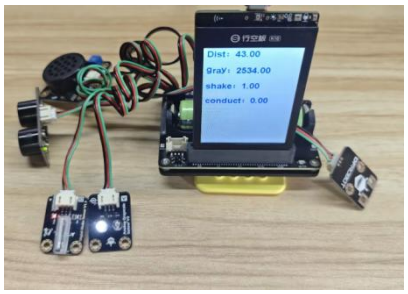
倒车雷达：将手在超声波传感器前慢慢靠近，喇叭播放 DADADADUM。

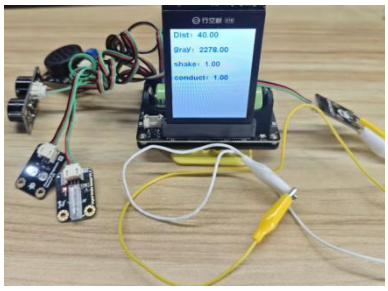
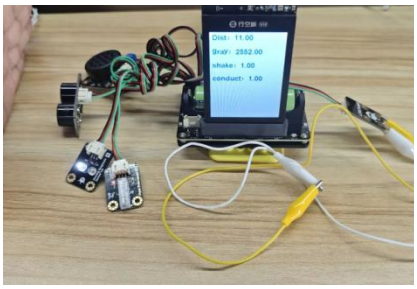
车道偏离：将灰度传感器对准黑色表面，喇叭播放 PRELUDE。

涉水检测：用湿手指触碰电导开关的两个触点，喇叭播放 ENTERTAINER。

碰撞检测：摇晃整个装置，喇叭播放 ODE。

如果传感器判断不灵敏，调整程序中的阈值后重新上传测试。



知识园地

四合一传感器融合——智能汽车的"五官"

智能汽车依靠多种传感器感知环境，本项目用四个传感器模拟了其中的四种：

超声波测距 → 倒车雷达（模拟输入）

超声波传感器发出一束人耳听不到的声波，遇到障碍物反射回来，通过计算发射到接收的时间差算出距离。就像蝙蝠的回声定位一样。

灰度传感器 → 车道偏离预警（模拟输入）

红外 LED 照射地面，接收管检测反射光强度。白色路面反射强（数值小），黑色车道线吸收光（数值大）。智能小车就是靠这个原理"寻迹"行驶的。

电导开关 → 涉水检测（数字输入）

检测两个触点之间是否有导电物质（水）连通电路。下雨天路面有积水时，车辆通过检测水的导电性来判断是否进入涉水路段。

晃动传感器 → 碰撞检测（数字输入）

内部有一个金属滚珠或弹簧结构，晃动时滚珠跳动瞬间导通电路。用来模拟车辆发生碰撞或剧烈颠簸时的紧急检测。

多传感器融合（Multi-Sensor Fusion）

单个传感器只能感知环境的某一个维度——超声波只知道距离，灰度只知道颜色。但如果把它们的数据综合起来，就能得到更丰富的判断：

距离近 + 晃动 → "追尾了！"

灰度数值骤变 → "车辆偏离车道！"

距离近 + 灰度骤变 + 导电 = 1 → "涉水路段有障碍物！"

这就是多传感器融合的核心理念：**1+1>2**，多个传感器的组合能产生单个传感器无法得出的结论。

拓展挑战

1. **融合判断**：同时满足两个以上警报条件时（如距离近+晃动），屏幕显示"追尾危险！"的综合警告。
2. **驾驶舱仪表盘**：在屏幕上用图形或图标的方式显示四个传感器的状态，设计一个更有科技感的界面。

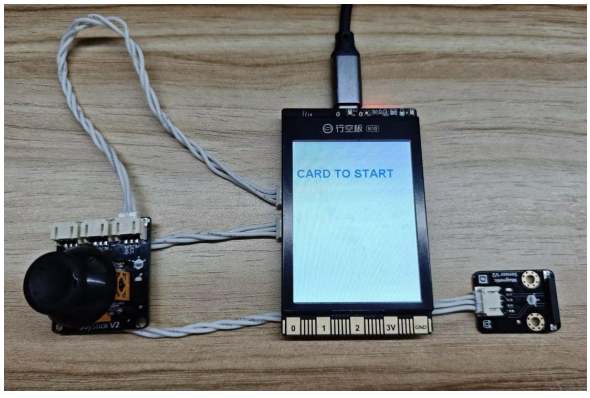
第 9 课 摇杆反应挑战

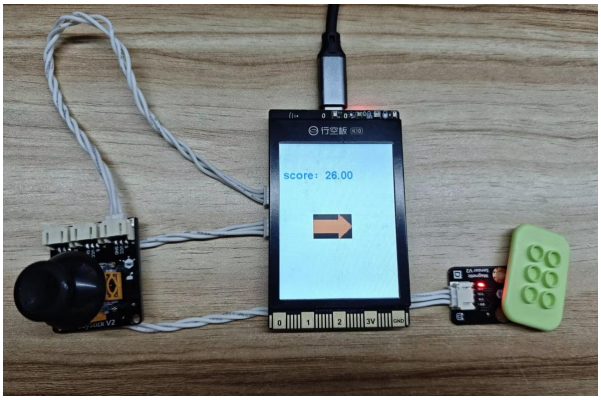
你玩过反应力测试游戏吗？屏幕上随机出现左右箭头，你要在最短时间内推动摇杆匹配方向，磁感应传感器刷卡开始游戏。

摇杆反应挑战就是一个融合了方向操控和磁卡开关的趣味项目。摇杆匹配屏幕箭头方向，磁感应传感器刷卡开始游戏。

任务目标

设计并制作一个摇杆反应挑战：摇杆匹配屏幕随机显示的左右方向，磁感应传感器刷卡开始/重新开始游戏。





学习目标

- 学会使用摇杆模块读取 X 轴数值判断推动方向

- 学会使用数字磁感应传感器检测磁场
- 理解游戏编程中的状态转换和逻辑判断

材料清单

硬件清单

行空板 K10(DFR0992) x 1

JoyStick 摇杆(DFR0061) x 1

数字贴片磁感应传感器(DFR0033) x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

本项目主要通过下面两个任务，逐步完成摇杆反应挑战的制作与调试。在任务一中，学会读取摇杆 X 轴数值并确定左右方向判断阈值。在任务二中，加入磁感应传感器实现刷卡控制游戏状态，完成完整的游戏。

任务一：读取摇杆 X 轴数值

读取摇杆模块的 X 轴模拟值，在屏幕上显示，观察推动摇杆时数值的变化规律，确定左右方向的判断阈值。

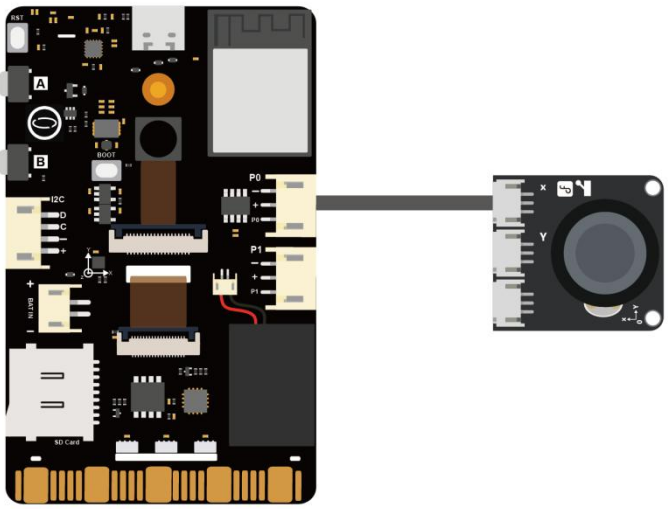
任务二：刷卡控制游戏状态

在任务一的基础上加入磁感应传感器，实现刷卡开始游戏、摇杆匹配方向加分的完整演示。

任务一：读取摇杆 X 轴数值

1. 硬件连接

将摇杆模块连接到行空板 K10 的 **P0** 端口。



2. 软件准备

1. 打开 **Mind+** 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**

3. 程序编写

首先使用"设置引脚 模式"指令，将 P0 设为"INPUT"输入模式。

创建"变量 x"，用来存放摇杆 X 轴读取到的模拟值，并在主程序开始下初始化该变量的值为 0。

在“重复执行”中，使用“读取模拟引脚 P0”指令，获取摇杆 X 轴的模拟值，并将数据赋值给“变量 x”。

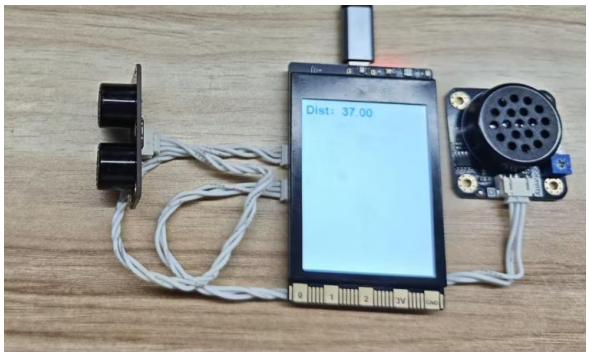
最后，使用“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，将缓存中的数据显示。

4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。

推动摇杆，观察 X 轴数值变化（向左推接近 0，向右推接近 4095，居中约 2048）。

记下摇杆向左和向右时的数值范围，为后续任务设置判断阈值做准备。

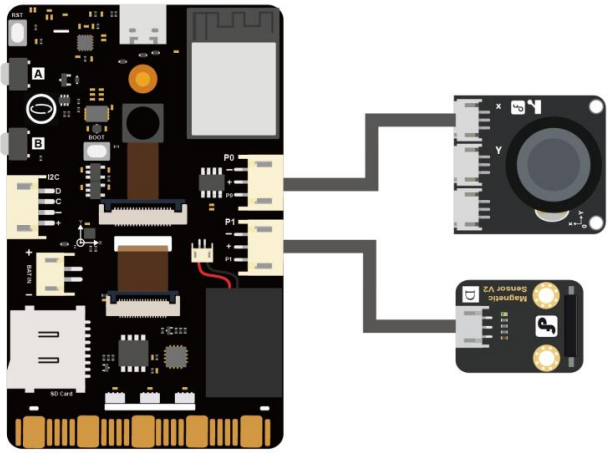


任务二：刷卡控制游戏状态

在任务一的基础上加入磁感应传感器，实现刷卡开始游戏、摇杆匹配方向加分的简单演示。

1. 硬件连接

保持任务一的硬件连接不变，将磁感应传感器连接到行空板 K10 的 P1 端口。



2. 程序编写

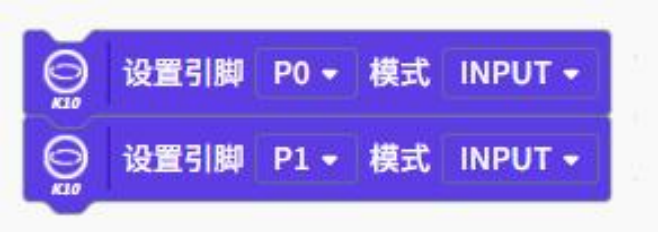
游戏只有两个状态：**等待开始**（刷卡启动）和**游戏中**（摇杆匹配箭头）。

新建以下变量：

- x：摇杆 X 轴读取到的模拟值
- magnet：磁感应传感器读取到的数字值（0 或 1）
- score：当前得分
- target：目标方向（1=左，2=右）
- game_state：游戏状态（0=等待开始，1=游戏中）

变量创建成功后，初始化所有变量的值为 0。

使用"设置引脚 模式"指令，将 P0 设为"INPUT"输入模式，P1 设为"INPUT"输入模式。



在**重复执行**中，先读取模拟引脚 P0 获取摇杆 X 轴数值，赋值给变量 x。再读取数字引脚 P1 获取磁感应传感器数值，赋值给变量 magnet。

接下来，进行游戏状态判断，当变量 game_state = 0（等待开始），使用“缓存显示文字 ”和“将缓存内容显示/显示更新”指令，在屏幕上显示"CARD TO START"。

在等待开始的状态中，如果磁感应传感器检测到磁铁时（读取数字引脚 P1 = 1），将变量 game_state 设为 1。

使用“随机数 ”指令，生成随机数 1 或 2 并赋值给变量 target，生成一个随机的方向图片。

当变量 game_state = 1（开始游戏），使用“缓存显示文字 ”和“将缓存内容显示/显示更新”指令在屏幕上显示当前得分。

游戏状态中，当变量 target = 1，使用“缓存显示本地加载图片 在坐标 X70 Y150”与“将缓存内容显示/显示更新”指令，在屏幕上显示向左的箭头。

当变量 target = 2，使用“缓存显示本地加载图片 在坐标 X70 Y150”与“将缓存内容显示/显示更新”指令，在屏幕上显示向右的箭头。

当摇杆向左推（X < 1800）且目标为左（变量 target = 1）时，摇杆推动的方向与屏幕上显示箭头方向一致，得分+1（将 score 增加 1）。等待 300ms 后生成新的随机目标（设置 target 的值为 1-2 之间的随机数），再等待 200ms 防止连续加分。

当摇杆向右推（X > 2200）且目标为右（变量 target = 2）时，摇杆推动的方向与屏幕上显示箭头方向一致，得分加 1（将 score 增加 1），等待 300ms 后生成新的随机目标（设置 target 的值为 1-2 之间的随机数），再等待 200ms 防止连续加分。

完整程序如下：

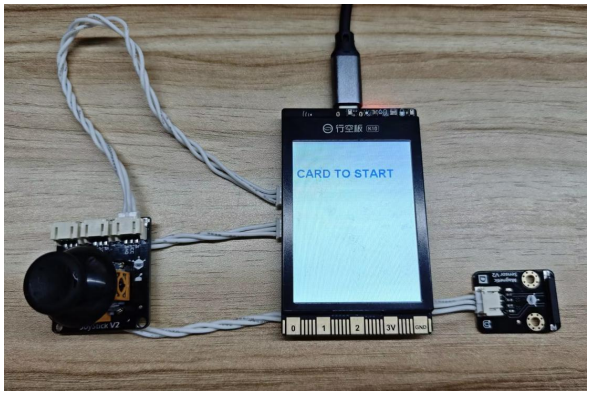

行空板K10主程序开始
设置 X 的值为 0
设置 magnet 的值为 0
设置 score 的值为 0
设置 target 的值为 0
设置 game_state 的值为 0
设置引脚 P0 模式 INPUT
设置引脚 P1 模式 INPUT
循环执行
设置 X 的值为 读取模拟引脚 P0
设置 magnet 的值为 读取数字引脚 P1
如果 变量 game_state = 0 那么执行
缓存显示文字 "CARD TO START" 在第 3 行 颜色
将缓存内容显示/显示更新
如果 变量 magnet = 1 那么执行
设置 game_state 的值为 1
设置 target 的值为 在 1 和 2 之间取随机数
如果 变量 game_state = 1 那么执行
缓存显示文字 合并 "score:" 变量 score 在第 3 行 颜色
将缓存内容显示/显示更新
如果 变量 target = 1 那么执行
缓存显示本地加载图片 在坐标X 70 Y 150
将缓存内容显示/显示更新
如果 变量 target = 2 那么执行
缓存显示本地加载图片 在坐标X 70 Y 150
将缓存内容显示/显示更新
如果 变量 X < 1800 与 变量 target = 1 那么执行
将 score 增加 1
等待 0.3 秒
设置 target 的值为 在 1 和 2 之间取随机数
等待 0.2 秒
如果 变量 X > 2200 与 变量 target = 2 那么执行
将 score 增加 1
等待 0.3 秒
设置 target 的值为 在 1 和 2 之间取随机数
等待 0.2 秒

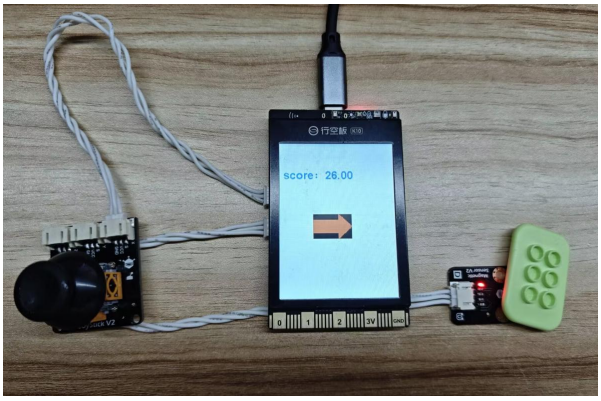
3. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。

- **开始游戏**：刷卡后屏幕随机显示"←"或"→"。
- **游戏操作**：摇杆向左推匹配"←"，向右推匹配"→"，正确得分+1 并换新箭头。
- **推错方向**：不扣分不结束，箭头保持不变。

如果摇杆判断不灵敏，调整程序中的阈值（1800 和 2200）后重新上传测试。





知识园地

摇杆——方向控制的桥梁

摇杆模块内部集成了两个电位器（X 轴和 Y 轴）。推动摇杆时，对应轴的电阻值变化，主控板读取到的模拟电压也随之改变：

- 居中时约 2048（中间值）
- 推到最左接近 0
- 推到最右接近 4095

游戏中通过设定左右阈值（1800 和 2200），将连续的模拟值转换为明确的"左"或"右"方向判断。

磁感应传感器——游戏开关

贴片磁感应传感器内部是一个干簧管。磁铁靠近时内部金属片磁化吸合，电路导通输出高电平；磁铁远离后断开输出低电平。游戏中用它作为开始/重新开始的触发开关，简单可靠。

游戏状态机

游戏程序采用了**状态机**的编程思想——把游戏过程分为两个清晰的状态（等待开始→游戏中），不同状态下程序对同一输入做出不同响应。这是一种非常实用的编程模式，常用于游戏、自动售货机、交通灯等场景。

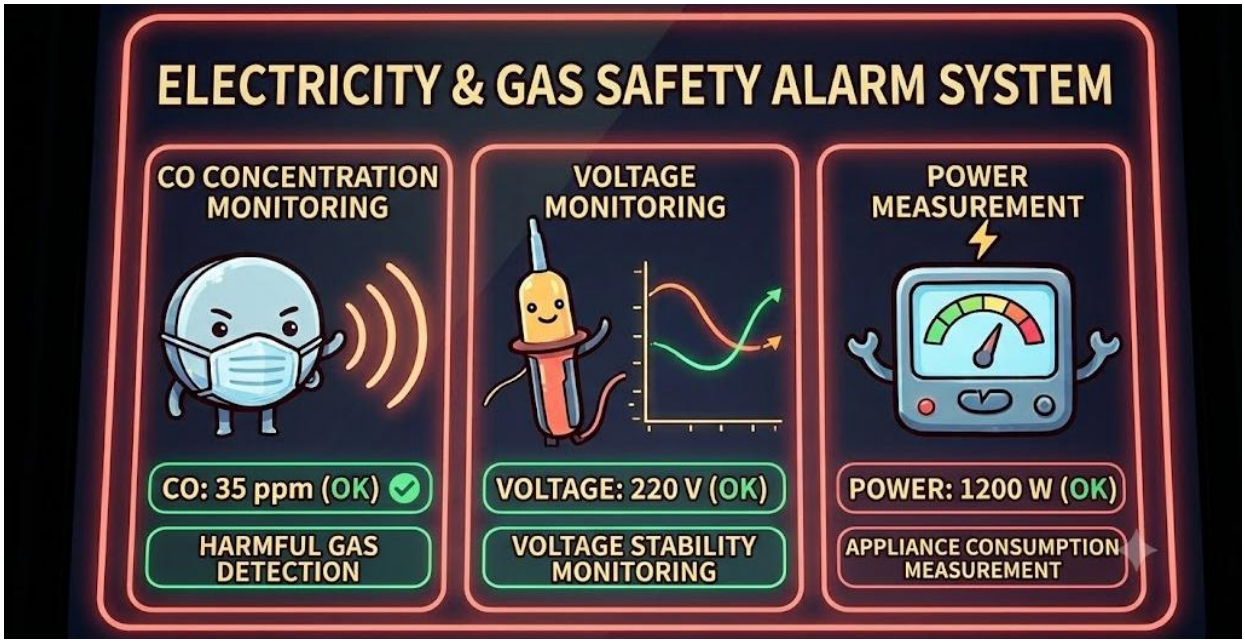
拓展挑战

1. **难度递增**：每得 5 分后缩短等待时间（从 0.2 秒逐渐减少到 0.05 秒），让游戏越来越快。
2. **组合得分**：每连续答对 5 次额外奖励 1 分，答错时连续记录清零。
3. **心率监测挑战（需额外接入心率传感器 SEN0203）**：将心率传感器连接到 P0 端口，在游戏过程中实时监测心跳。当检测到心率超过 120 时，游戏结束。

第 10 课 气电安全报警仪

厨房里的燃气热水器有没有泄漏？家里的电压稳不稳定？电器到底消耗了多少电？这些问题关系到我们的生活安全。如果有一个小设备能同时监测一氧化碳浓度、电压和功率，发现问题时 LED 闪烁报警，那该多省心。

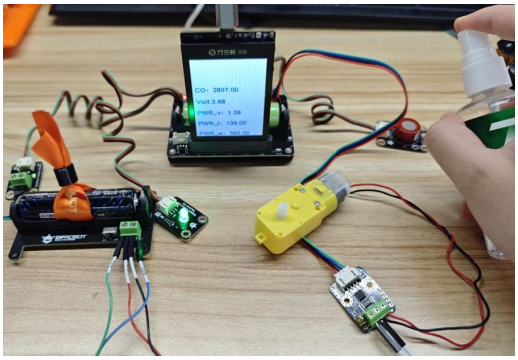
气电安全报警仪就是这样一个"安全管家"。本项目将使用一氧化碳气体传感器检测有害气体、电压检测模块监测电压、数字功率计测量功率，浓度或电压超标时 LED 闪烁报警。



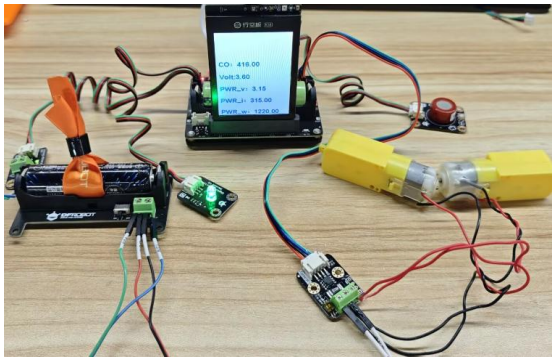
任务目标

设计并制作一个气电安全报警仪：实时检测一氧化碳浓度、电压值和功率数据，超阈值时 LED 闪烁警报。

CO 异常



功率异常



学习目标

- 学会使用 MQ7 一氧化碳气体传感器检测 CO 浓度

- 学会使用模拟电压检测模块测量电压
- 学会使用 I2C 数字功率计读取功率数据

材料清单

硬件清单

行空板 K10(DFR0992) x 1

行空板多功能扩展板 (DFR1216) x 1

模拟一氧化碳气体传感器 (MQ7)(SEN0132) x 1

模拟电压检测模块（DFR0051） x 1

I2C 数字功率计（SEN0291） x 1

LED 模块（DFR0021） x 1

软件使用

Mind+编程软件（V1.8.1 RC3.0 及以上版本）

下载地址：https://mindplus.cc/

动手实践

本项目主要通过下面三个任务，逐步完成气电安全报警仪的制作与调试。在任务一中，制作一个 CO 气体报警器——检测到一氧化碳浓度超标时 LED 闪烁。在任务二中，加入电压检测模块，扩展电压异常报警功能。在任务三中，加入数字功率计，实现完整的综合安全监测。

任务一：CO 气体报警器

读取 MQ7 气体传感器的一氧化碳浓度值，超过阈值时 LED 快速闪烁报警。

任务二：电压异常报警器

在 CO 报警器基础上加入电压检测模块，实现有害气体和电压异常双重报警。

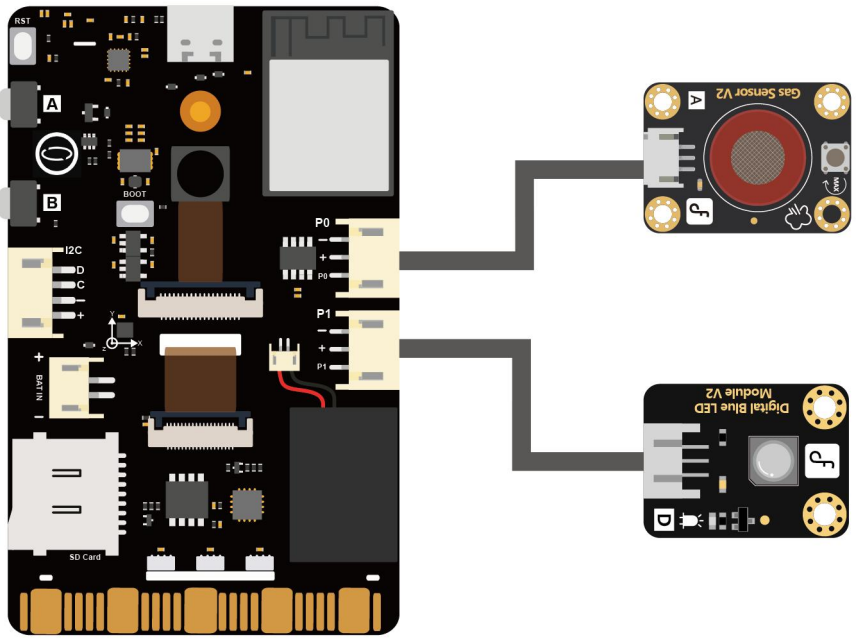
任务三：综合安全监测系统

加入数字功率计读取电参数，三种传感器同时工作，LED 根据异常类型以不同节奏闪烁。

任务一：CO 气体报警器

1. 硬件连接

将 CO 气体传感器连接到行空板 K10 的 **P0** 端口，LED 连接到 **P1** 端口。



2. 软件准备

1. 打开 **Mind+** 软件
2. 选择 **上传模式**
3. 在"主控扩展"中选择 **行空板 K10**

3. 程序编写

首先使用"设置引脚 模式"指令，将 P0 设置为"INPUT"输入模式，P1 设置为"OUTPUT"输出模式。

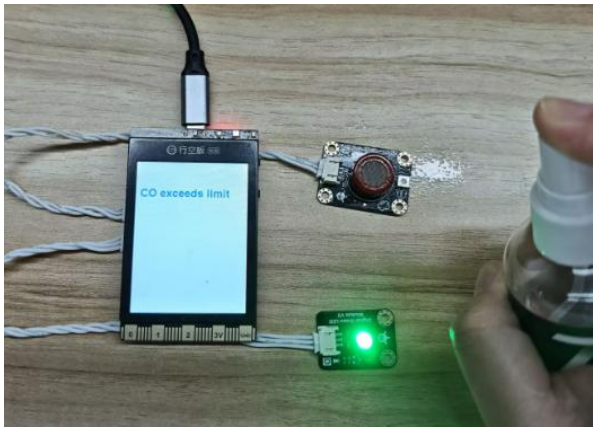
创建“变量 CO”，并初始化变量的值为 0。在重复执行中，使用“读取模拟引脚 P0 ”指令，读取 CO 浓度数值，并将读取的值存放在变量 CO 中。

当数值大于 2000 时（变量 CO > 2000），认为一氧化碳超标。使用“设置数值引脚 P1 输出高电平/低电平”指令，控制 LED 以急促节奏（0.2 秒间隔）闪烁报警。

最后使用“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，在屏幕上显示"CO 超标！"。当数值正常时 LED 熄灭，显示"CO 正常"。完整程序如下：

4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。观察屏幕上的 CO 数值，在传感器附近轻轻吹气，观察数值上升。当数值超过 2000 时，LED 快速闪烁，屏幕显示"⚠CO 超标！"

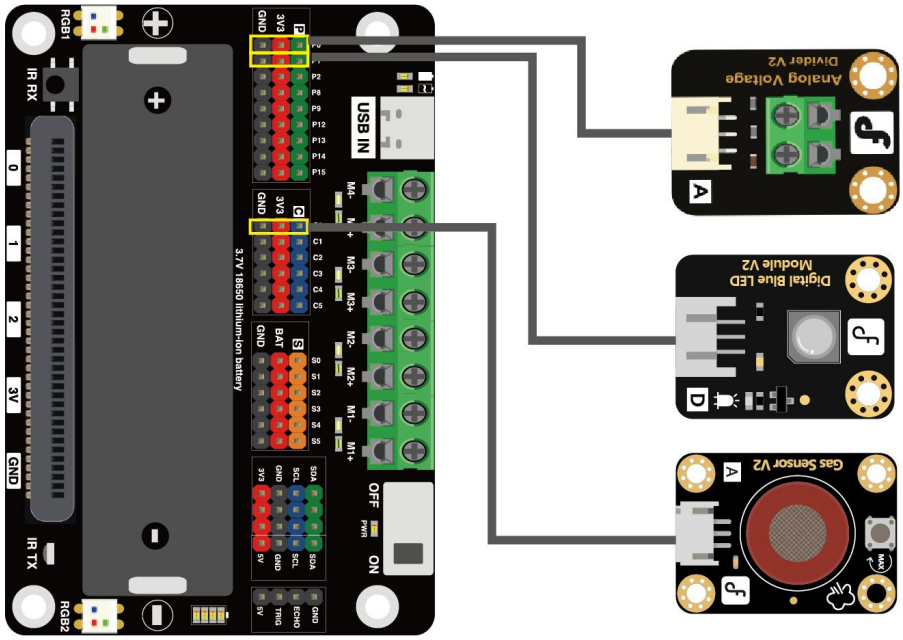


任务二：电压异常报警器

在任务一的基础上，加入电压检测模块，实现 CO 超标和电压异常双重报警功能。

1. 硬件连接

任务二需要使用**行空板多功能扩展板 DFR1216** 来扩展接口。将 CO 气体传感器连接到扩展板的 **C0** 端口，电压检测模块连接到行空板 K10 的 **P0** 端口，LED 连接到扩展板的 **P1** 端口。



2. 软件准备

在任务一的基础上：

1. 在"模块扩展"中搜索 **DFR1216**，下载并添加行空板多功能扩展板扩展库
2. 在"模块扩展"中搜索 **DFR0051**，下载并添加模拟电压检测模块扩展库

3. 程序编写

创建“变量 CO”、“变量 volt”，并初始化两个变量的值为 0。

在重复执行中，使用“读取引脚 C0 的模拟值”指令与“读取引脚 P0 电压值”指令，同时读取 CO 浓度和电压值，并将获取的数值存放到对应的变量中。

注意：使用扩展板时，C 口和 P 口对应不同的库指令——C 口的模拟值需用 **DFR1216 扩展库**的"读取引脚 C0 的模拟值"指令获取，电压值用 **DFR0051 扩展库**的"读取电压值"指令获取（返回单位 V），P 口的数字输出使用 **行空板 K10** 自带的"设置数字引脚输出"指令即可。

当检测到 CO 浓度超标时（变量 CO > 2000），通过“设置数值引脚 P1 输出高电平/低电平”指令，控制 P1 引脚的 LED 灯快闪报警。

当检测到电压异常偏低时（变量 volt < 1.5），通过“设置数值引脚 P1 输出高电平/低电平”指令，控制 P1 引脚的 LED 灯慢闪报警。

最后使用“缓存显示文字 ”与“将缓存内容显示/显示更新”指令，在行空板 K10 屏幕上显示检测到的对应数值。完整程序如下：

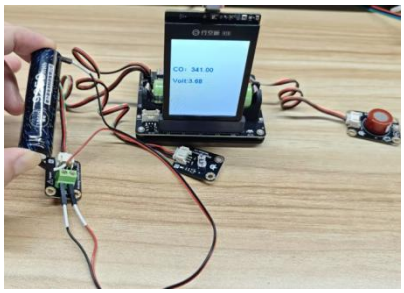
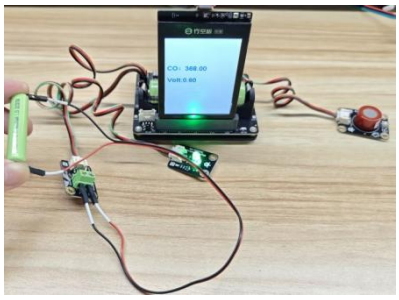
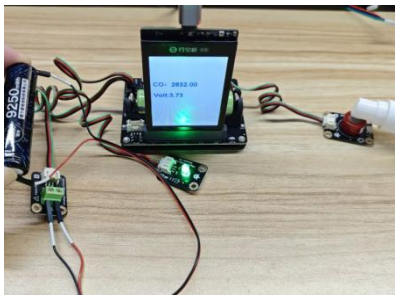
3. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。观察两种警报灯光的不同节奏。

CO 报警测试：在 CO 传感器附近吹气，浓度超标时 LED 快速闪。

电压异常测试：调整电压检测模块的输入，电压值低于 1.5V 时 LED 慢闪。

无异常　　电压异常　　CO 异常

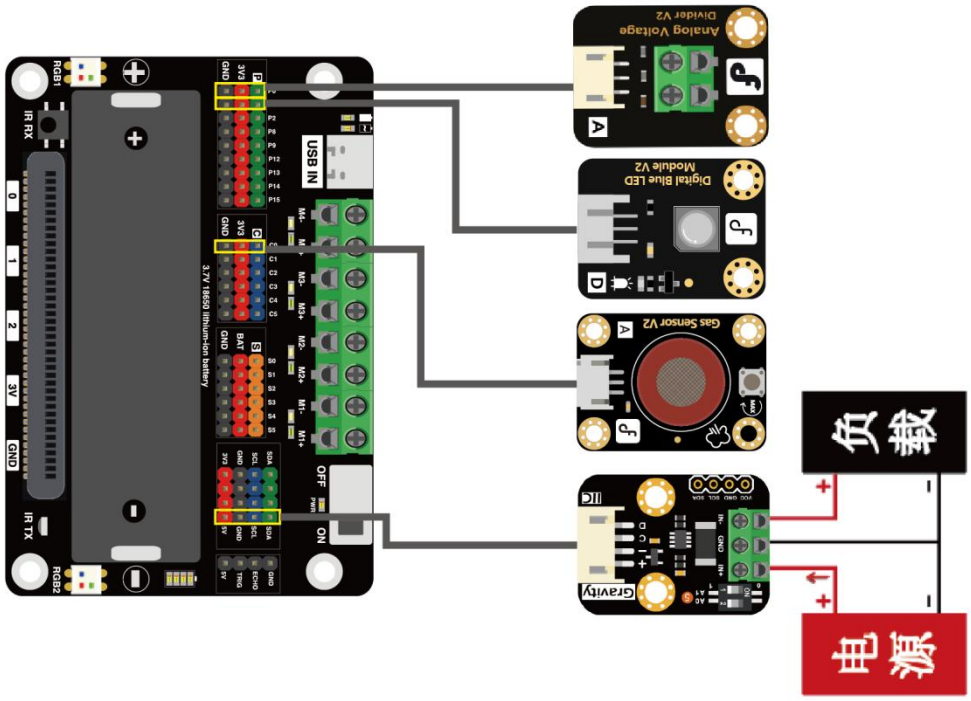
任务三：综合安全监测系统

在任务二的基础上，加入数字功率计，实时读取电压、电流和功率数据，三种传感器同时工作，LED 根据异常类型以不同节奏闪烁。

1. 硬件连接

任务三加入了数字功率计，它使用 I2C 接口，不占用 3Pin 端口。CO 气体传感器保持连接在扩展板的 **C0** 端口，电压检测模块保持连接在 **P0** 端口，LED 保持连接在扩展板的 **P1** 端口不变，将数字功率计连接到行空板 K10 的 **I2C 接口**。

注意：功率计模块需要串联接入用电设备的电路中，才能测量其电压、电流和功率，这需要对设备电路进行改造。功率计的详细使用方法，建议了解即可，具体可参考功率计模块的 wiki。



2. 软件准备

在任务二的基础上，在"模块扩展"中搜索 **SEN0291**，下载并添加。

3. 程序编写

在任务二已读取 CO 浓度和电压值的基础上，本任务只需再读取功率计的电压、电流和功率数据。

首先，使用“功率计 初始化 I2C 地址为 0x45”指令，初始化功率计，新建变量“变量 pwr_v ”、"变量 pwr_i"、"变量 pwr_w"，并初始化变量的值为 0。

在"重复执行"中，使用"功率计 读取负载电压/电流/功率"指令，将获取的数值存放到"变量 pwr_v"、"变量 pwr_i"、"变量 pwr_w"中。

注意：功率计数据需使用 **SEN0291 扩展库**的指令获取，读取的是串联接入电路中的电压、电流和功率，因此这里获取的电压是分流电压。

CO 超标和电压异常时的 LED 报警逻辑与任务二保持一致，本任务新增功率过大报警：当变量 pwr_w > 1000 时，通过"设置数值引脚 P1 输出高电平/低电平"指令，控制 P1 引脚的 LED 灯急闪报警（0.1 秒亮+0.1 秒灭）。

最后使用"缓存显示文字"与"将缓存内容显示/显示更新"指令，在行空板 K10 屏幕上分多行显示 CO 浓度、电压值以及功率计的电压、电流和功率数据。

三种警报灯光节奏各不相同：CO 超标为快速闪（快闪），电压异常为慢闪（警告），功率过大为紧急闪（急闪）。不同节奏让使用者单凭灯光就能判断是哪种警报。完整程序如下：

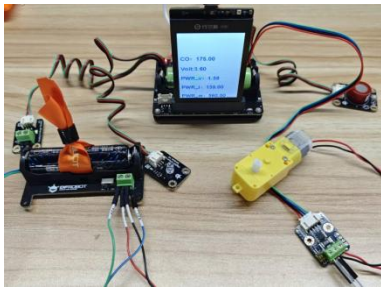
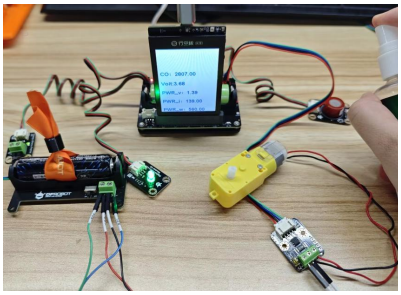
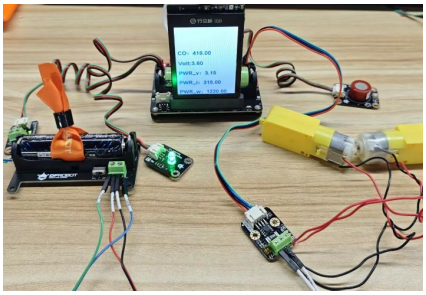
4. 程序运行

点击右上角"上传"按钮，将程序上传到行空板 K10。观察屏幕上一共五组数据：CO 浓

度、电压检测值、分流电压、电流、功率。

- **CO 报警**：在 CO 传感器附近吹气，数值超过 2000 时 LED 快速闪
- **电压异常**：电压检测数值低于 1.5V 时，LED 慢闪
- **功率过大**：在功率计接口接入负载，功率超过 1000 时 LED 急闪

无异常 CO 异常 功率异常

知识园地

MQ7 一氧化碳传感器——电"鼻子"

MQ7 是一种电化学气体传感器，内部有一个加热丝和敏感材料层。当一氧化碳分子与敏感材料接触时，材料的电阻值会发生改变，浓度越高电阻变化越大。

MQ7 需要先加热才能工作——上电后有大约 30 秒到 1 分钟的预热时间，这段时间数值会上下波动。使用时要远离明火和有机溶剂，也不要长时间在极高浓度下工作。

电压检测与功率计

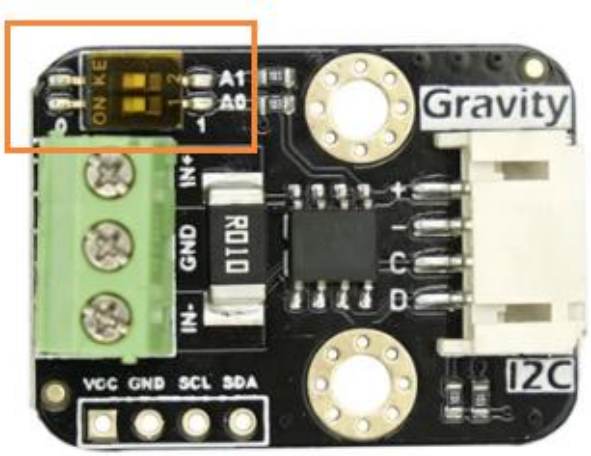
电压检测模块本质上是一个电阻分压器。它将待测的高电压按比例缩小到行空板 K10 可以安全读取的 0~3.3V 范围。通过读取模拟值并乘以分压比，就能换算出实际的电压值。

数字功率计（SEN0291）则更智能——它内部集成了电流检测芯片和运算电路，通过 I2C 协议直接输出电压、分流电压、电流和功率数据，无需手动换算。

三种传感器刚好体现了三种信号类型：CO 传感器是**模拟输入**，LED 是**数字输出**，功率计是 **I2C 通信**。

功率计的 I2C 地址——拨码开关的秘密

任务三中使用了"功率计 初始化 I2C 地址为 0x45"指令，这个地址可不是随便填的，它和功率计模块上的**拨码开关**一一对应。



功率计模块有一个 2 位拨码开关（标有 A0、A1）。通过拨动开关组合，可以把模块配置成 4 个不同的 I2C 地址：

拨码开关（A0 / A1）	I2C 地址
0 / 0	0x40
1 / 0	0x41
0 / 1	0x44
1 / 1	0x45（默认）

拨码开关拨到"1"一侧表示 1，另一侧表示 0。

所以程序里初始化地址 **0x45**，对应的正是拨码开关出厂默认的 A0=1、A1=1。**程序中的地址必须与模块拨码开关的状态一致**，否则功率计无法被识别，屏幕上读不到电压、电流和功率数据。

这个设计也方便了多模块扩展：如果在一根 I2C 总线上挂多个功率计，只要把每个模块的拨码开关拨成不同组合，让它们使用不同的地址，就能互不干扰地同时工作。

分级报警——灯光也有信息

项目中三种警报使用了不同的灯光节奏：

- **快速闪烁**（0.2 秒间隔）：紧急情况，需要立即处理
- **慢速闪烁**（0.5 秒）：警告情况，需要关注
- **急闪**：突发情况，需要检查

这就是人机交互中的"视觉编码"——用不同的灯光模式传递不同的信息。

拓展挑战

1. **分页显示**：屏幕空间有限，让数据分页轮流显示（第一页 CO+电压，第二页功率+电流）。
2. **多级报警**：为 CO 设置两个阈值——超过阈值 1 时 LED 慢闪，超过阈值 2 时 LED 快闪。
3. **报警锁定**：报警触发后即使数值恢复正常，LED 仍持续亮，需要手动按键复位。