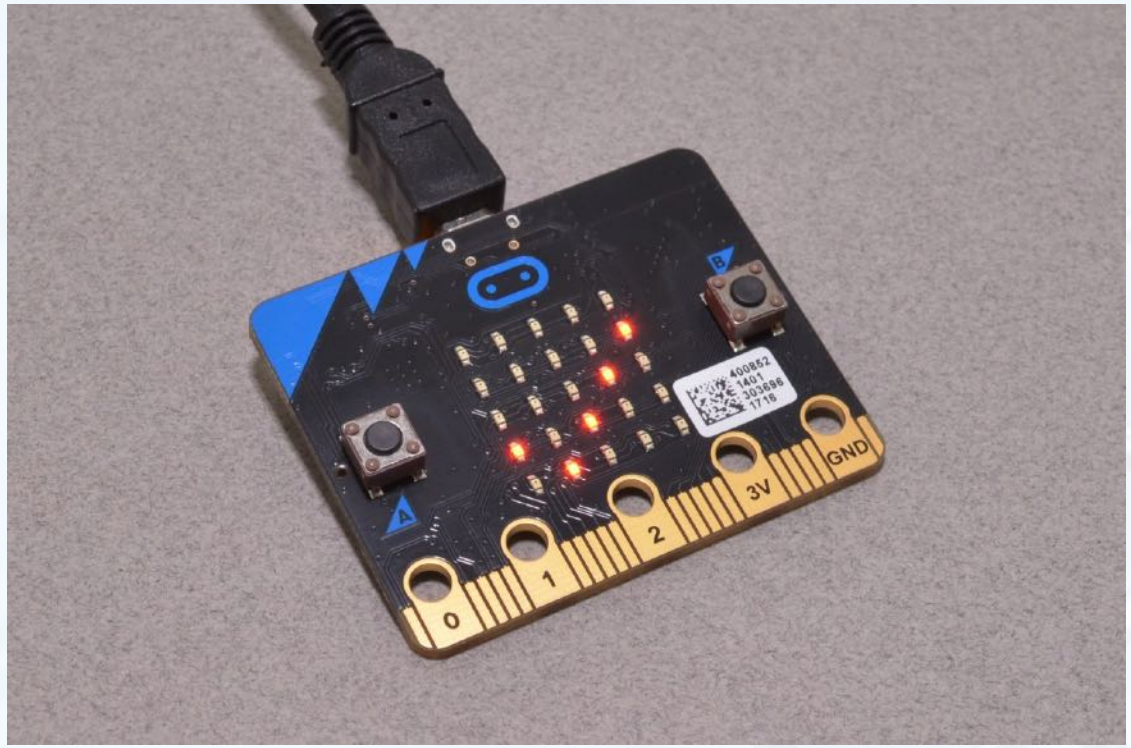
数据结构与算法（Python）-microbit

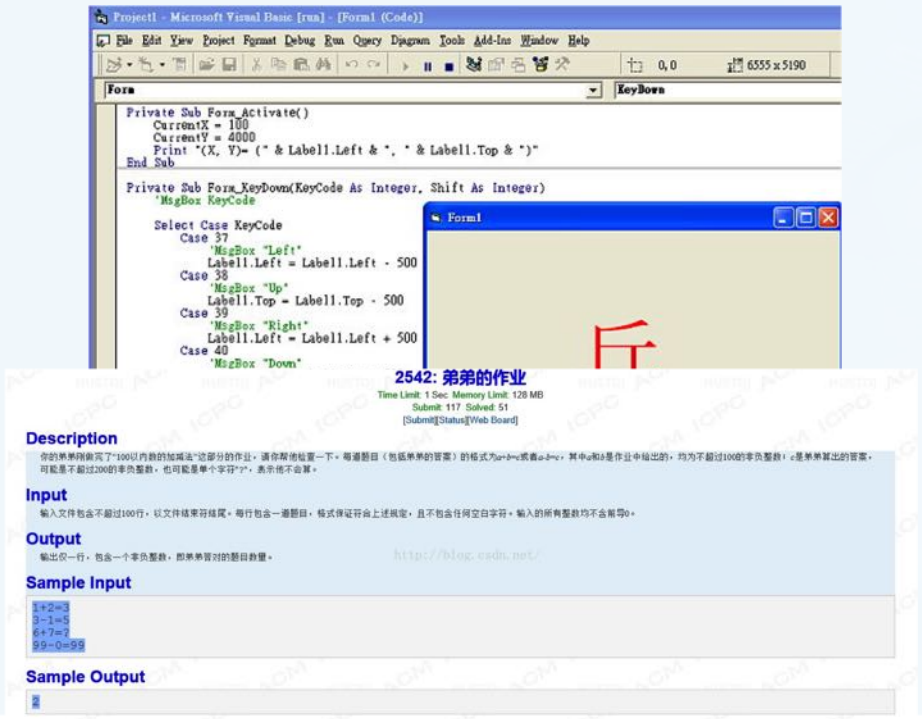
陈斌 gischen@pku.edu.cn 北京大学地球与空间科学学院

目录

- 开源硬件教学创新
- 初识microbit-micropython
- microbit硬件规格
- microbit基础编程



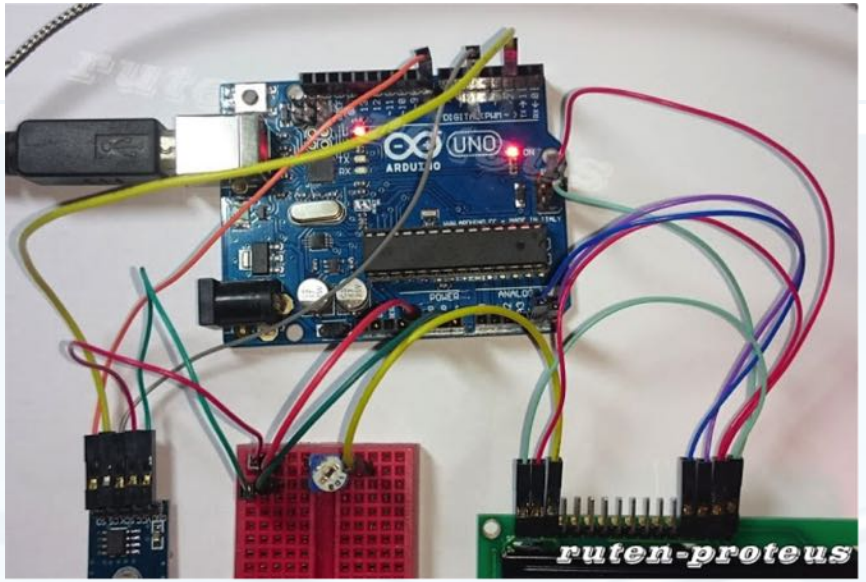
传统编程语言教学



- 以VB，C/C++等算法语言为主
- 偏向数学问题、字符处理等，程序结果反馈的形式单一
- 通常是用程序解数学应用题，抽象程度高，较为枯燥
- 通过OJ形式练习和考试，不友好
- 除了少数信息竞赛生，对多数学生吸引力不高

传统开源硬件教学

- 以arduino、51等单片机为主
- 用C语言编程，门槛较高
- 杜邦线、面包板连接，需要熟悉基本的电子电路知识，容易出错
- 程序逻辑与底层硬件端口访问混合，代码结构不清晰
- 适合少数科创竞赛学生

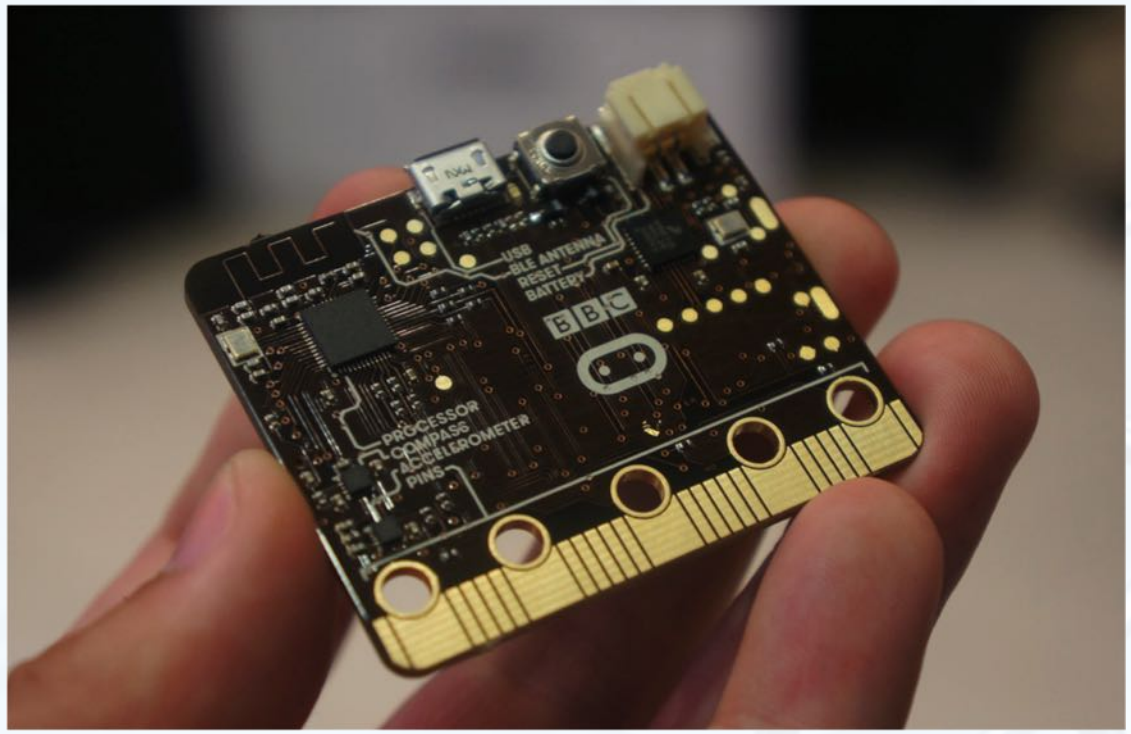



```
void loop() {
 // 確認 TC 是否開路或是故障
 if( is_tc_open )
 {
  static uint8_t tc_check;
  Serial.println( "TC is Open !");
  #ifdef USEI2CLCD
   displayCharOnLCD( 1, 1, "  TC is Open !  ", 16 );
   displayCharOnLCD( 2, 1, "                ", 16 );
   char a = 0x35 - (char)tc_check;
   displayCharOnLCD( 2, 8, &a , 1 );
  #endif

  // 每 5 秒鐘重新確認 TC 是否已重新連接上
  if( tc_check ++ > 3 )
  {
   max6675_getCelsius();
   tc_check = 0;
  }
 }
 else
 {
  float C = max6675_getCelsius();
  float F = max6675_getFahrenheit();
  Serial.print( C );
  Serial.println(" C");
  Serial.print( F );
  Serial.println(" F");
  #ifdef USEI2CLCD
   char buf[17];
```

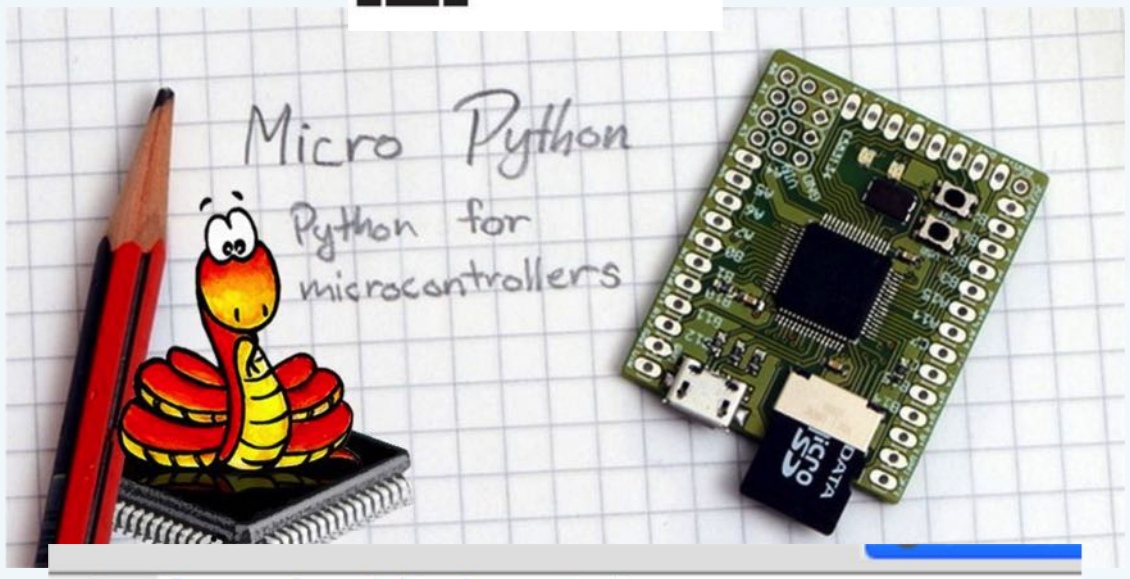
```
void clearLCD()
{

 Wire.beginTransmission(ADDRI2CLCD);

 Wire.write( 0x80 ); // One command
 Wire.write( 0x01 ); // Clear display

 Wire.endTransmission();
}

void displayCharOnLCD( int line, int column,
{
 unsigned char i;

 Wire.beginTransmission(ADDRI2CLCD);

 Wire.write( 0x80 );
 Wire.write( 0x80 + ( line - 1 ) * 0x40 + ( c
 Wire.write( 0x40 );

 for( i = 0; i < len; i++)
 {
  Wire.write( *dp++ );
 }

 Wire.endTransmission();
```

通过开源硬件进行编程语言教学

- 学习编程语言，而非学单片机
- 利用硬件的实物特性，声光电多种反馈形式，提高学习兴趣
- 需要消除单片机编程的繁杂
- 建立良好的层次抽象，凸显主要的程序逻辑
- 培养计算思维，编程解决问题
- Programming not coding



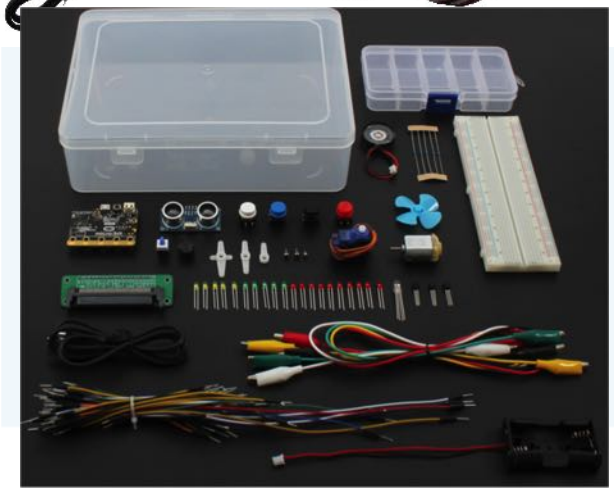
micropython：Python语言教学利器




```
1  from microbit import *
2
3  while True:
4      if accelerometer.was_gesture('shake'):
5          display.show(Image.HAPPY)
6          sleep(1000)
7          display.clear()
```

- 在单片机上直接使用Python语言编程
- 面向对象、动态、脚本语言的特性，使得访问硬件端口非常简单
- 有效隔离了底层繁琐的细节，对传感器、效应器等访问也非常便捷
- 大大降低单片机操作门槛，提高编程语言学习兴趣

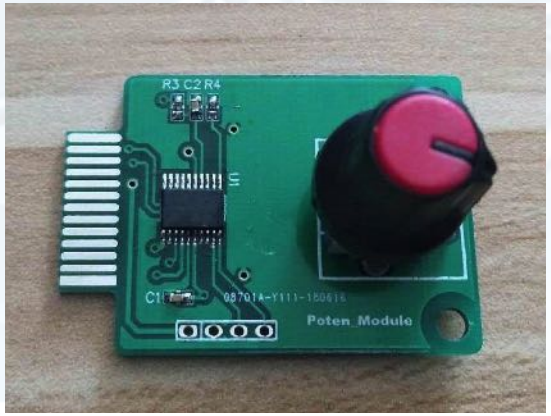
microbit的各种扩展板套件



- 多数仍然是杜邦线加面包板
- 杂乱不易打理
- 主要面向图形化编程，对Python语言支持度不高
- 对传感器抽象程度较低，仍然面向引脚读取数值

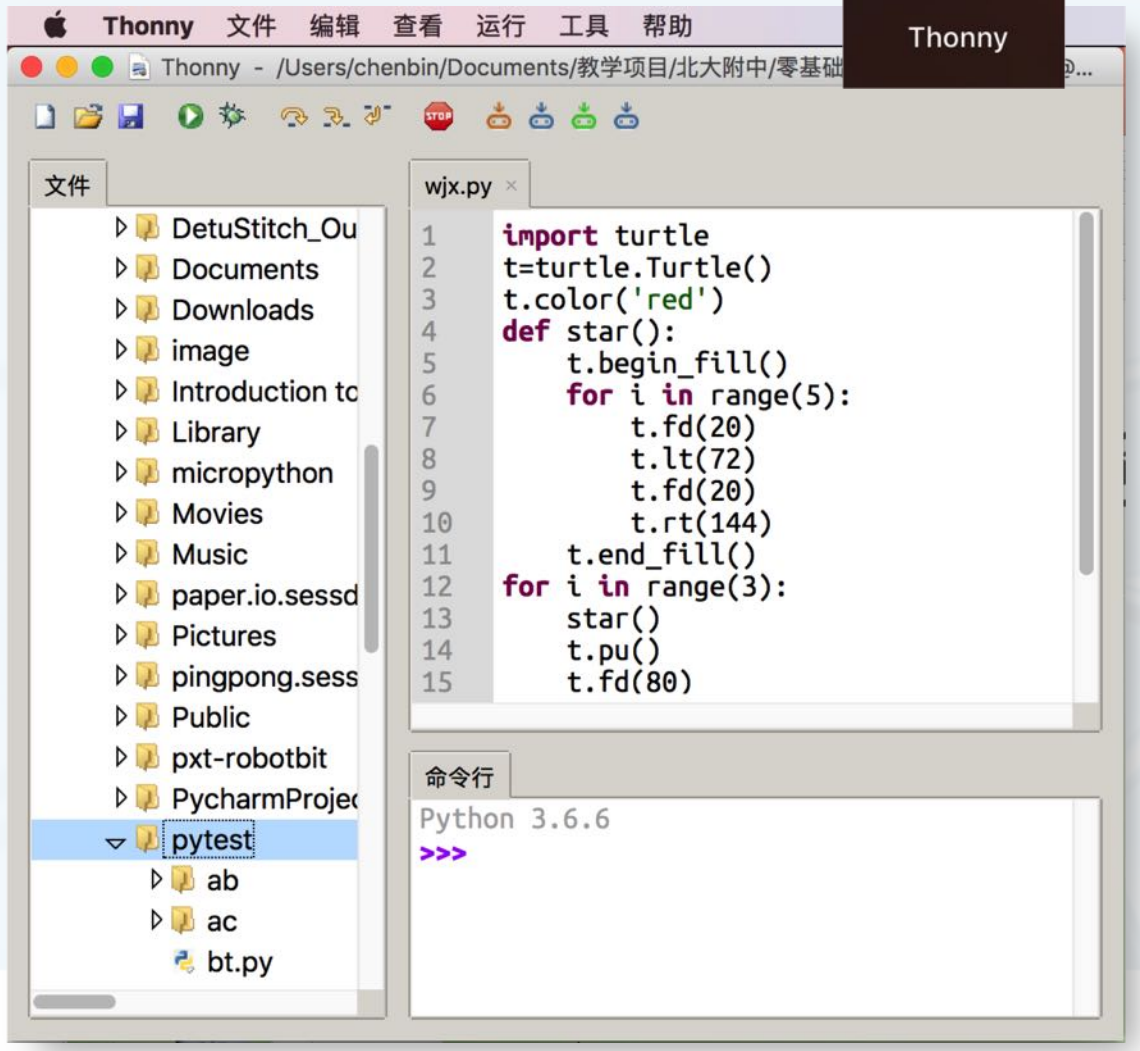
地小空开放实验室dxkStick扩展板套件

- 全插卡式传感器模块，消除杜邦线带来的杂乱困扰
- 完全面向Python语言编程教学
- 通过高层次指令访问传感器，程序逻辑清晰
- 支持多个单片机之间的分组无线通信
- 兼容乐高LEGO构件尺寸
- 开放的指令系统，易于扩展

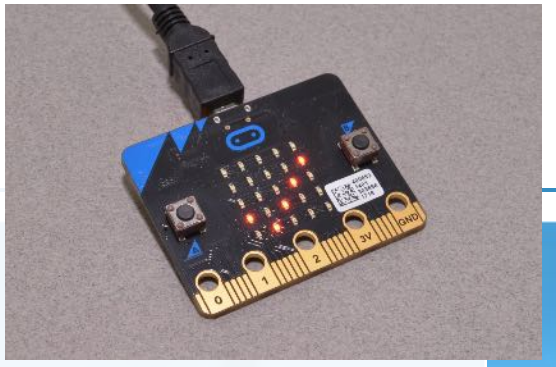


集成开发环境：Thonny

- 跨平台Windows/macOS/Linux
- 自带最新版本Python3，无需安装Python
- 体积小巧，功能齐全
- 安装第三方模块很方便
- 可以连接microbit单片机编程
- 地小空开放实验室汉化版本
  https://github.com/chbpku/dxkStickIDE/tree/master/Setup



初识microbit-micropython



- USB线连接microbit
- 打开Thonny
- 输入程序，保存为py文件
- 点击“写入运行环境”
  稍等一下，闪烁结束
  出现成功Done字样
- 点击“写入当前代码”
  出现成功Done字样
- 立刻运行，可按RESET重启

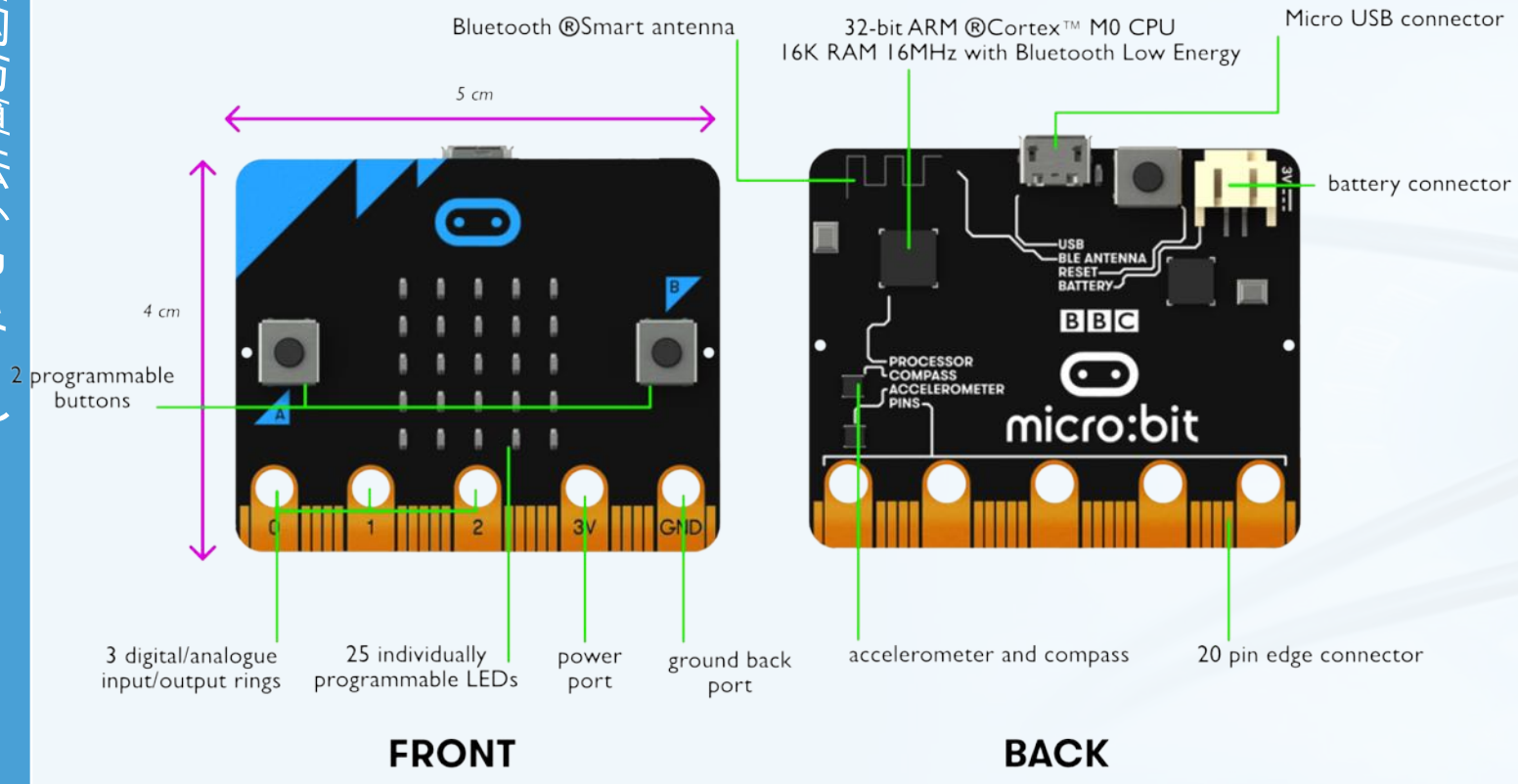
实例1：电子骰子

```
from microbit import *
from random import randint

while True:
    display.show('*')
    if accelerometer.was_gesture('shake'):
        display.clear()
        display.show(str(randint(1,6)))
        sleep(1000)
    sleep(10)
```

实例2：萤火虫

```
from microbit import *
from random import *
import radio

display.show(Image.HAPPY)
f = [Image().invert()*(i/9) for i in range(9, -1, -1)]

radio.on()

while True:
    if button_a.was_pressed():
        radio.send('flash')  # 按键发送呼叫信息
    r = radio.receive()
    if r == 'flash':
        sleep(randrange(1000)+50)
        display.show(f, delay=100, wait=False)
        if randrange(10) <= 0:  #回应的机会0-9，数字越大，机会越大
            sleep(500)
            radio.send('flash')
```

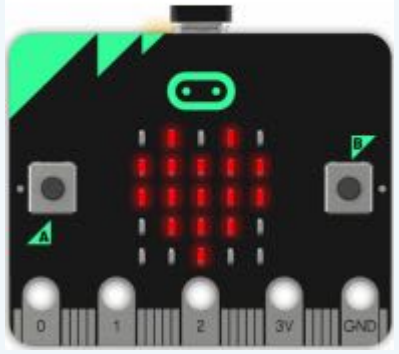
microbit from BBC介绍



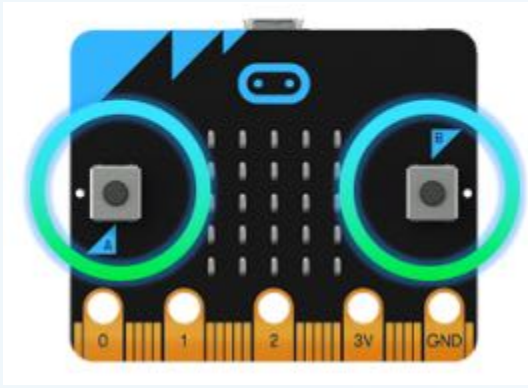
- 25个独立编程的LED
- 2个可编程的按钮
- 1个reset按钮
- microUSB接口
- 3V电源接口
- 光线传感器、温度传感器
- 加速计、电子罗盘
- 无线通信：射频以及蓝牙

microbit概貌

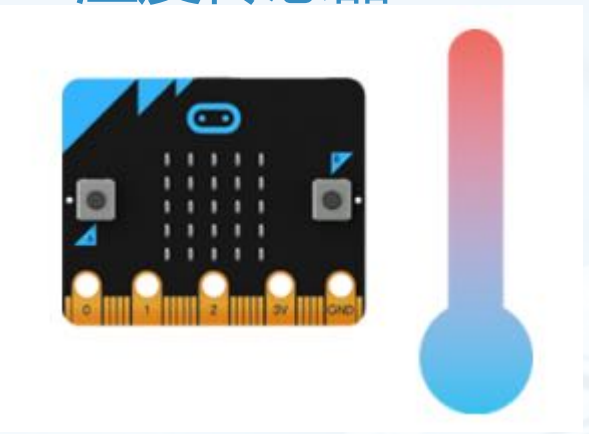
LED



按钮传感器



温度传感器

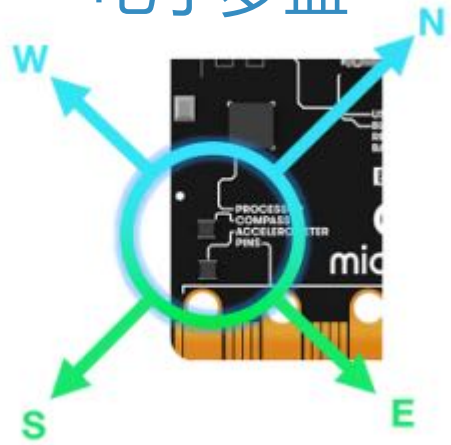


射频

光线传感器

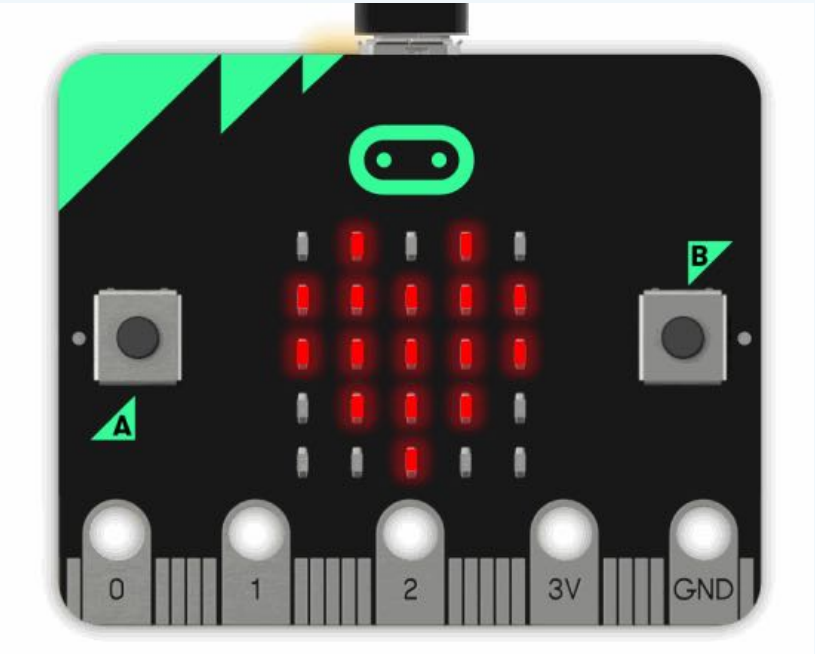
加速度计

电子罗盘



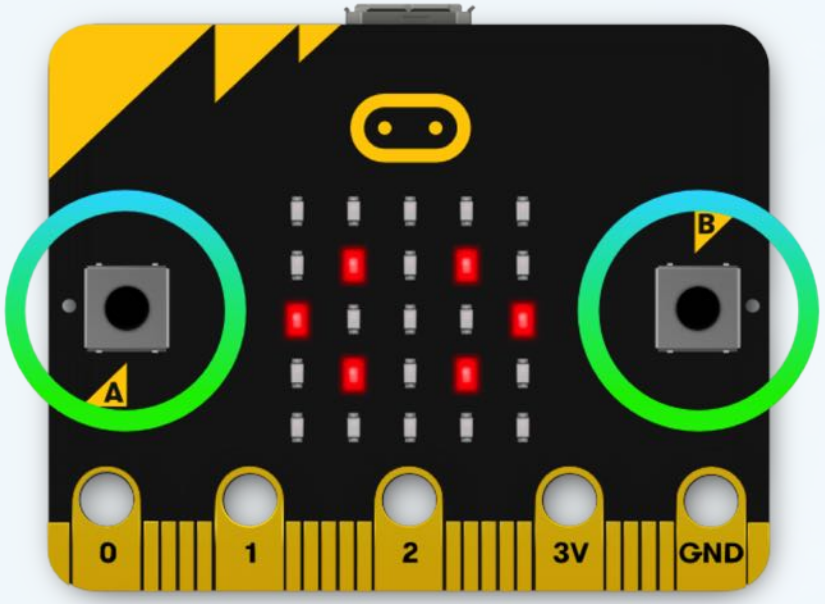
蓝牙

microbit特征：LED



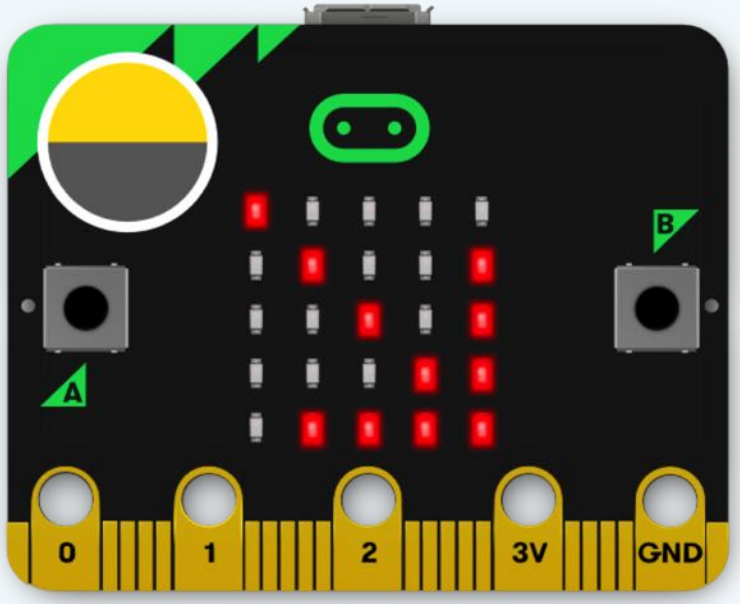
- micro:bit有25颗可独立编程的LED灯
- 可以用来显示文本，数字以及图像

microbit特征：按钮



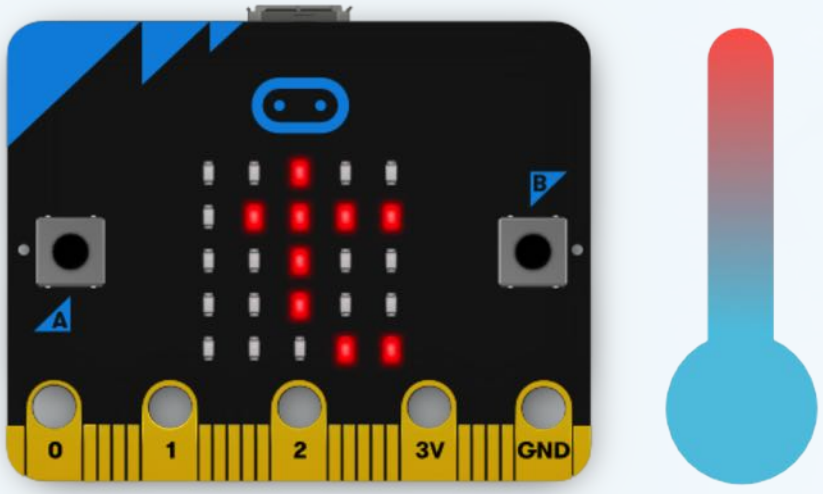
- 在micro:bit板子前面有2个按钮（标记了A和B）
- 可以检测按下这些按钮，运行代码
- 还可以检测这些按钮被按下的时间和次数

microbit特征：光线传感器



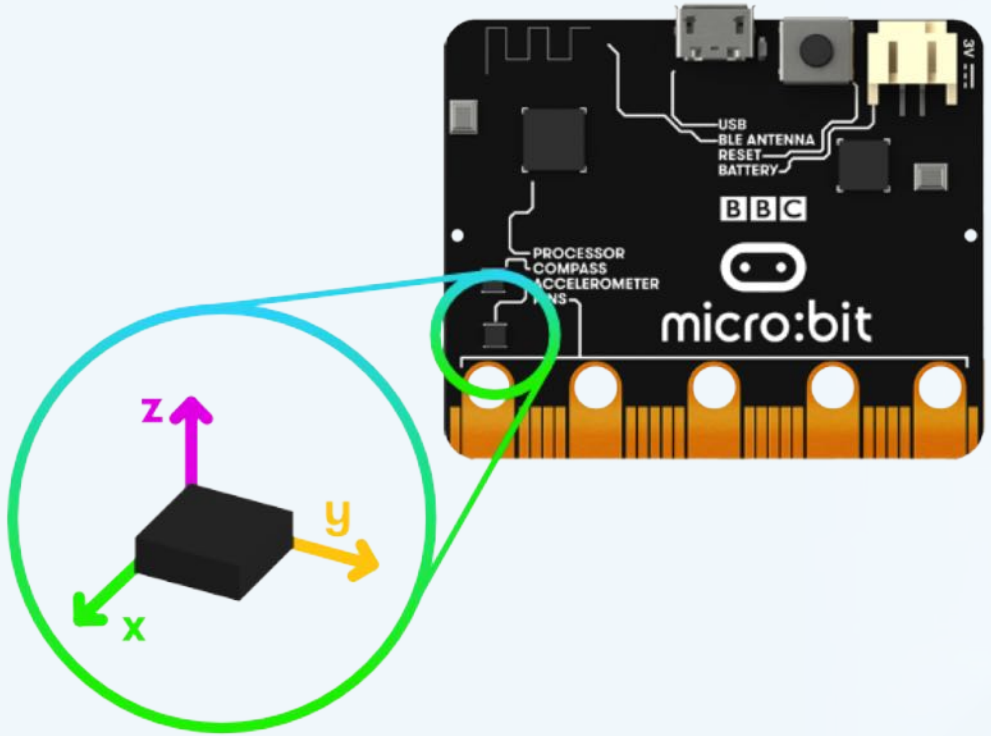
- 通过反转LED屏幕，micro:bit进入输入模式
- LED屏幕起到一个基础的光线传感器的作用
- 可以用来检测周围的光线

microbit特征：温度传感器



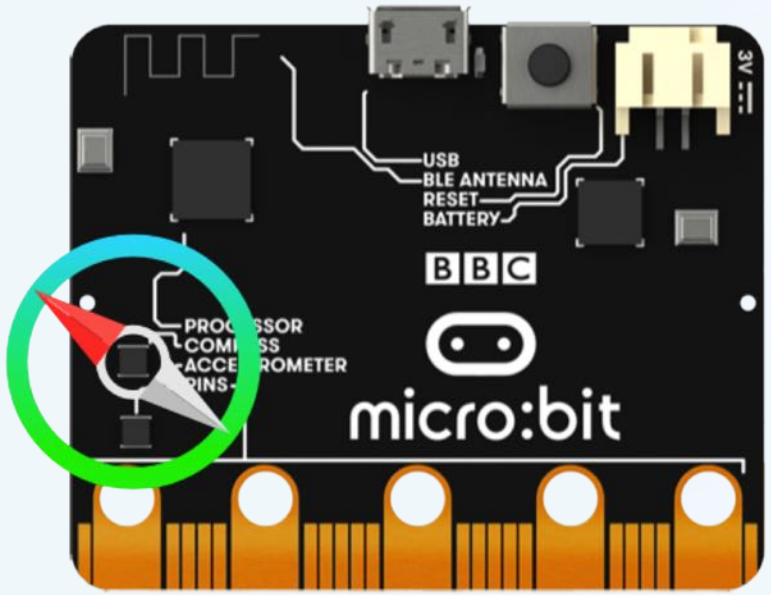
- 温度传感器可以让micro:bit检测当前环境温度(以摄氏度为单位)
- 温度传感器本来是用作检测处理器温度
- 如果处理器计算负载高的话，温度测量值也高

microbit特征：加速度传感器



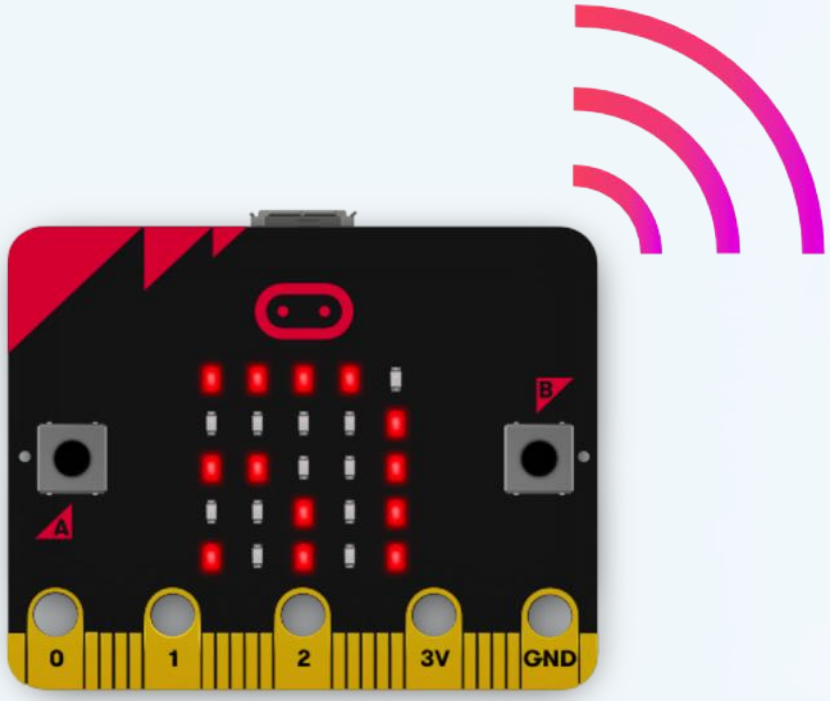
- 加速度传感器可以测量micro:bit的加速度
- 可以检测micro:bit的移动
- 也可以检测其他的动作
- 例如：摇动，倾斜以及自由落体。

microbit特征：指南针



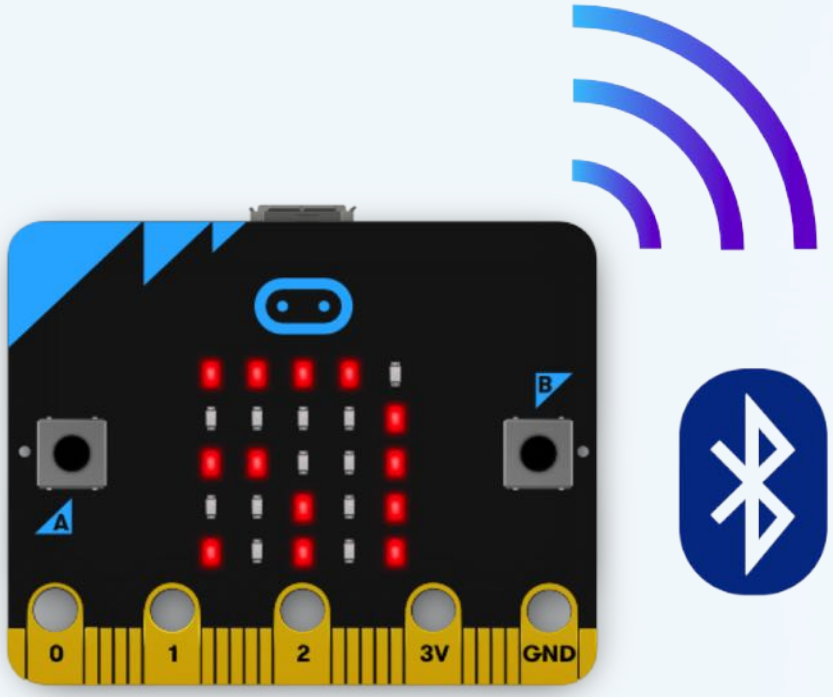
- 指南针用于检测地球磁场，可以探测到micro:bit面对的方向
- 在使用之前，需要校准指南针。
- “校准”是为了确保指南针的结果正确
- 在JavaScript积木块编辑器中，使用“指南针校准”积木块
- 在Python中用compass.calibrate()校准指南针

microbit特征：无线电



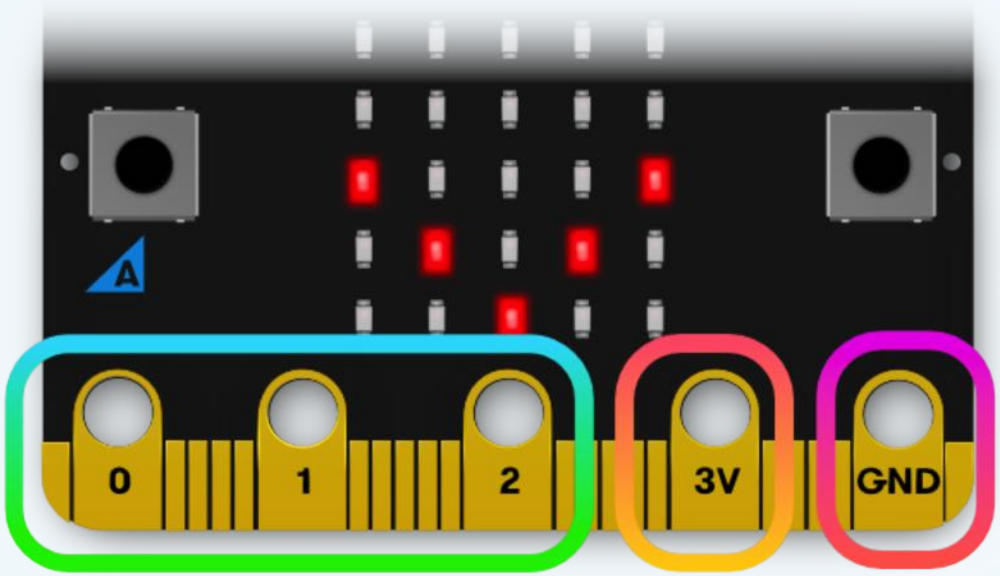
- 可以在2块甚至多块micro:bit板子之间进行无线通讯
- 用无线电发送信息到其他的micro:bit板子上
- 无线电可以有100个频道，调节发射功率，以免互相干扰
- 用于创建多人游戏以及更多有趣的发明！

microbit特征：蓝牙（Python不可用）



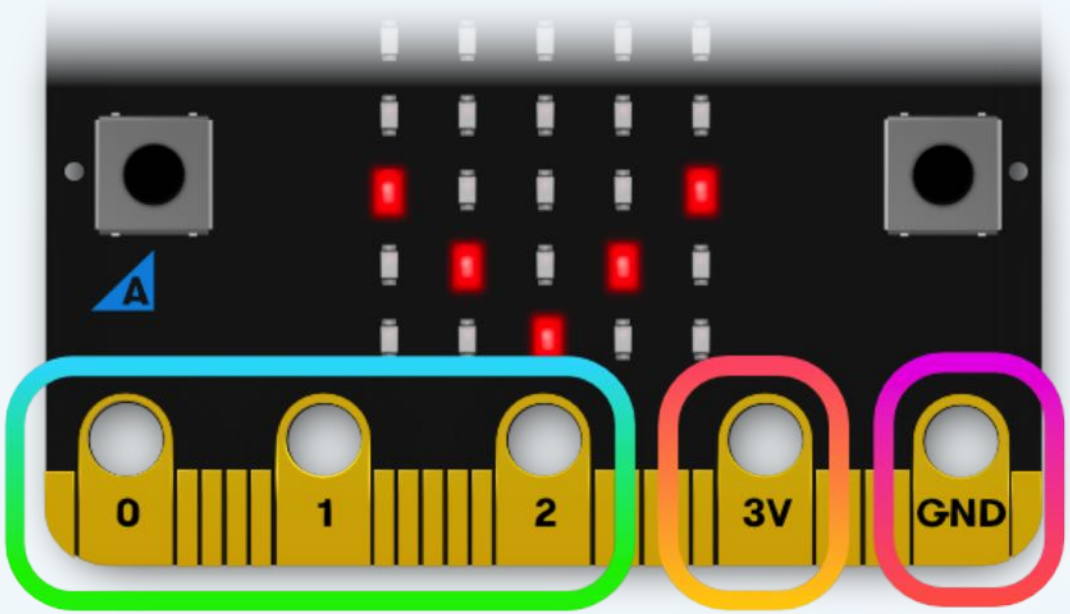
- BLE(蓝牙低能量）天线可以让micro:bit接收蓝牙信息
- 这可以让micro:bit和电脑，手机以及平板进行无线通信
- 可以用micro:bit控制手机
- 或者用手机发送无线代码到设备上

扩展引脚



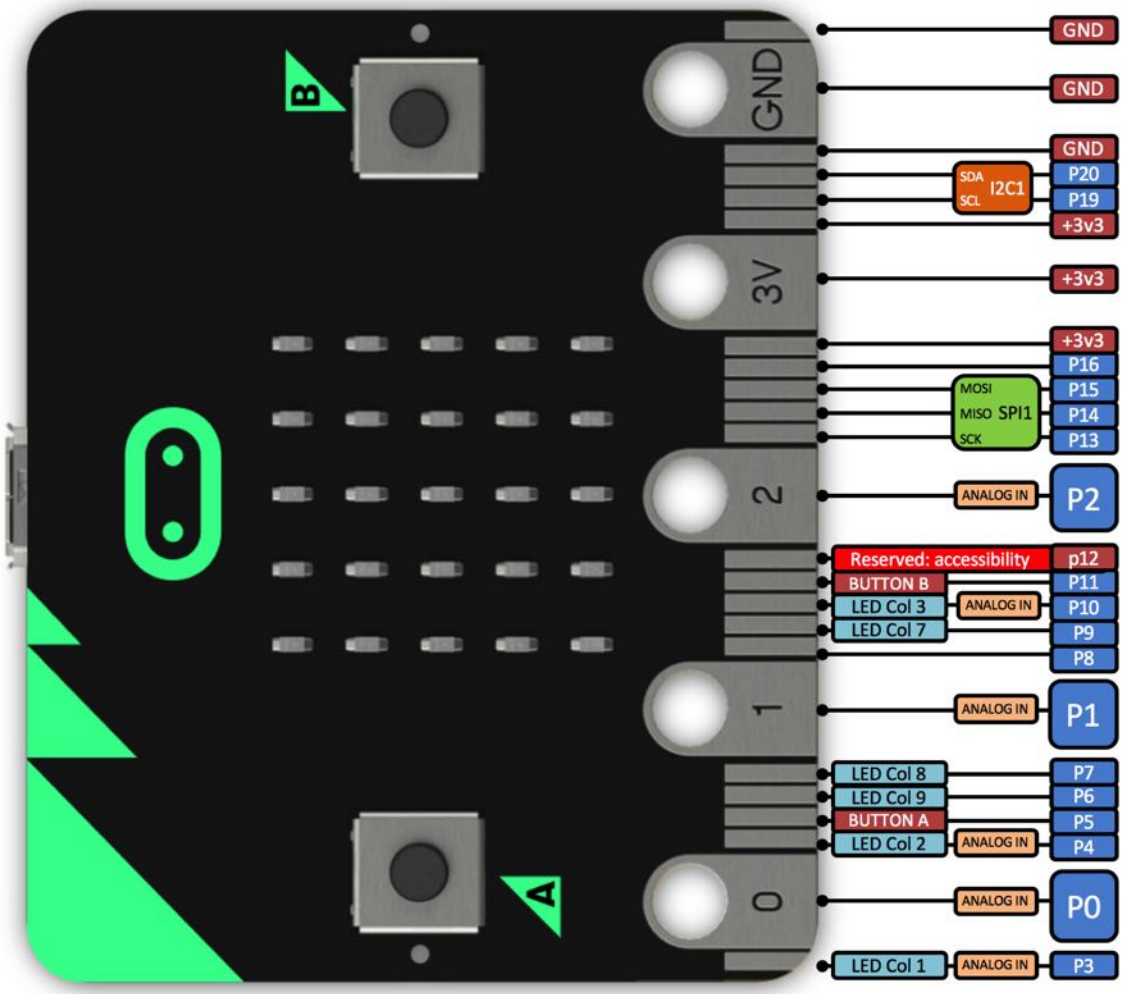
- 在micro:bit连接器的边缘有25个外部接口
- 这些接口称作“引脚”
- 引脚可以扩展连接电机，LED灯
- 或者其他带引脚的电子元器件编程
- 或者是连接外部传感器控制代码

microbit的扩展引脚：大引脚



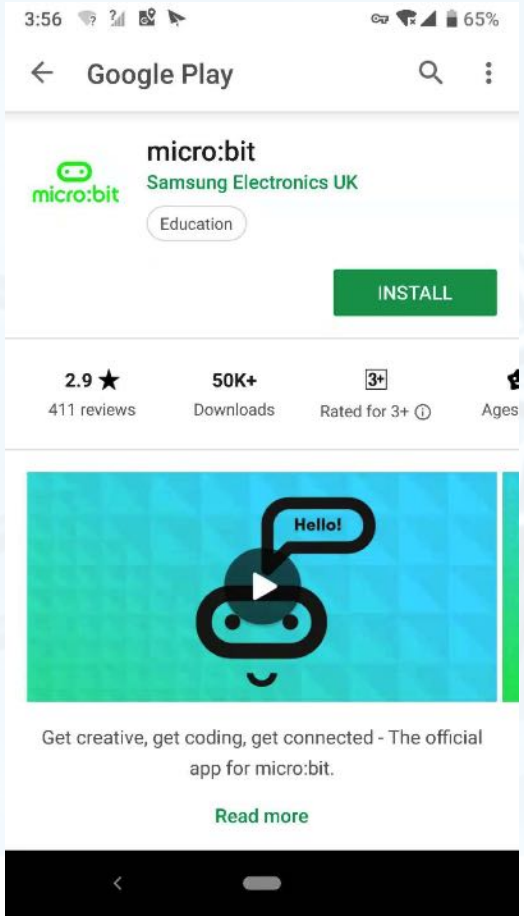
- P0 ： GPIO（通用数字输入和输出）带模拟-数字转换器 (ADC)
- P1: 带ADC的GPIO
- P2: 带ADC的GPIO
- 3V：电源
- GND：接地

microbit扩展引脚：小引脚



序号	名称	类型	作用
1	P0	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
2	P1	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
3	P2	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
4	P3	LED Col1/ADC-GPIO	如果关闭LED，可以当作ADC-GPIO用
5	P4	LED Col2/ADC-GPIO	如果关闭LED，可以当作ADC-GPIO用
6	P5	Button-A	按钮A
7	P6	LED Col9/D-GPIO	如果关闭LED，可以当作D-GPIO用
8	P7	LED Col8/D-GPIO	如果关闭LED，可以当作D-GPIO用
9	P8	D-GPIO	空闲的数字输入输出D-GPIO
10	P9	LED Col7/D-GPIO	如果关闭LED，可以当作D-GPIO用
11	P10	LED Col3/D-GPIO	如果关闭LED，可以当作ADC-GPIO用
12	P11	Button-B	按钮B
13	P12	D-GPIO	空闲的数字输入输出D-GPIO
14	P13	D-GPIO/SPI-SCK	缺省的SPI-SCK信号
15	P14	D-GPIO/SPI-MISO	缺省的SPI-MISO信号
16	P15	D-GPIO/SPI-MOSI	缺省的SPI-MOSI信号
17	P16	D-GPIO/SPI-NSS(CS)	空闲的数字输入输出D-GPIO，可用作SPI芯片选择
18	P17	3v3	电源输入
19	P18	3v3	电源输入
20	P19	I2C-SCL	I2C总线的时钟信号（内置加速计/指南针）
21	P20	I2C-SDA	I2C总线的数据线（内置加速计/指南针）
22	P21	GND	接地
23	P22	GND	接地

microbit的移动app



microbit图形编程和Javascript编程

- **快速入门**

  连接-编程-下载-操作

  https://microbit.org/zh-CN/guide/quick/

- **图形编程和Javascript编程工具**

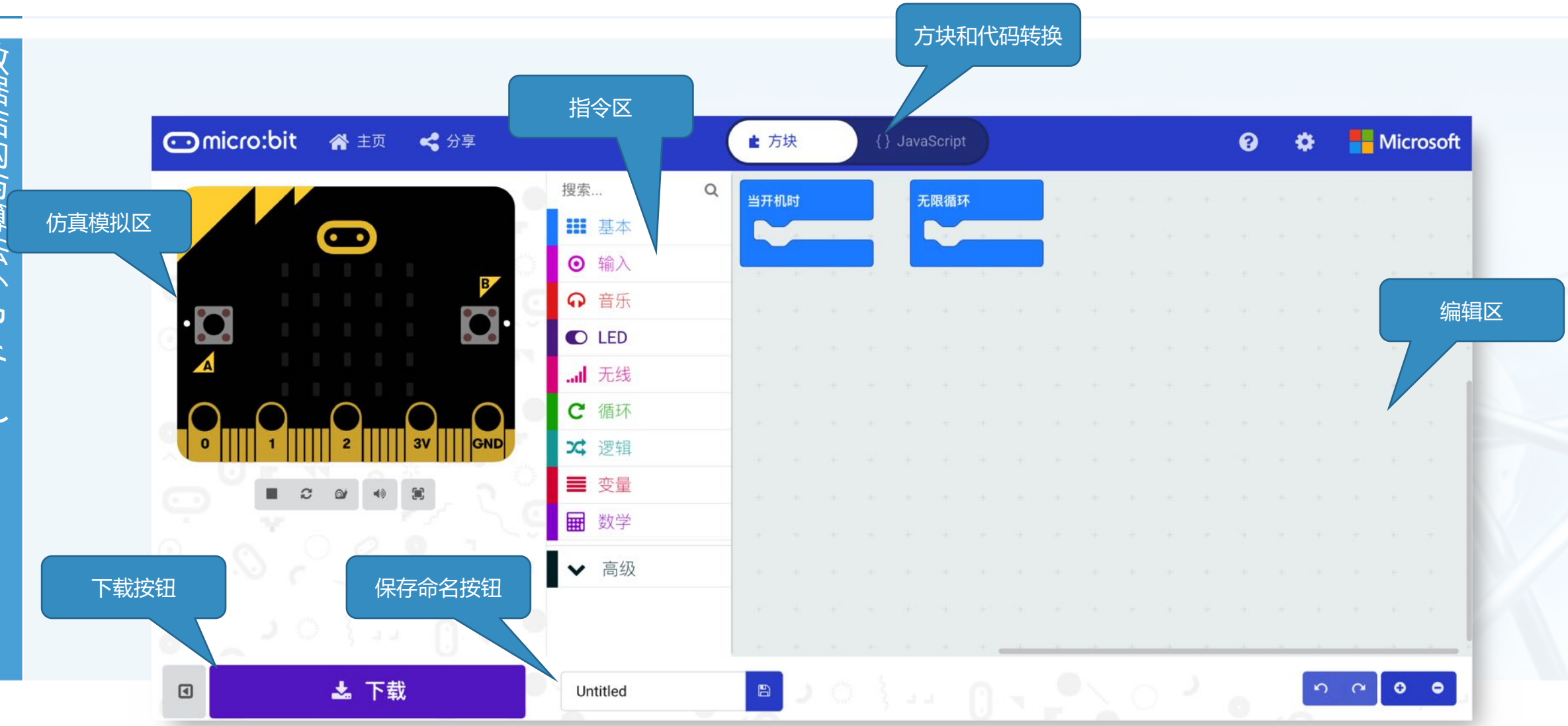
  由Microsoft的makecode项目提供

  https://makecode.microbit.org/#lang=zh-CN

- **在线Python编辑器**

  https://python.microbit.org/v/1.1

microbit图形编程界面



microbit基础编程体验：图形编程

教程

Flashing Heart

Name Tag

Smiley Buttons

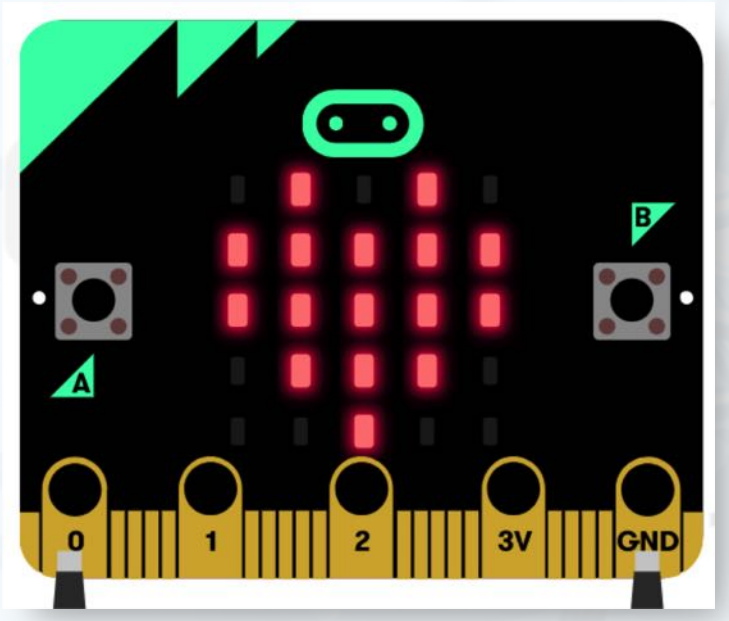
Dice

Love Meter

Micro Chat

microbit创意作品（2018）

- 课程面向同学们开展microbit创意作品开发活动
- 报名同学组成1-3人小组，从老师处领取一套microbit和扩展板
- 用Python语言开发出各式创意作品
- 同学们热情踊跃
- 由于microbit硬件数量的限制
- 共创作了34组作品



microbit创意作品人气榜的15个作品

① 多功能搬运车

② microbit模拟器

③ paper.io多人对抗游戏

④ 俄罗斯方块

⑤ 数算课的生死时速

⑥ 微钢琴

⑦ 遥控向日葵花车

⑧ microbit超级马里奥

⑨ 捕鱼达人

⑩ 宇宙飞船游戏系列

⑪ 音乐游戏MusicBlocks

⑫ 捣蛋机器人

⑬ 电子歌姬

⑭ 疯狂炸弹人

⑮ 音乐编辑器。

https://space.bilibili.com/275008758/#/favlist?fid=1702213

【sessdsa18】第07组：点球大战

收藏于5-7

【sessdsa18】第21组：奔跑计步器

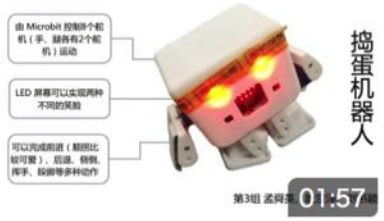
收藏于5-7

【sessdsa18】第12组：手速比拼

收藏于5-7

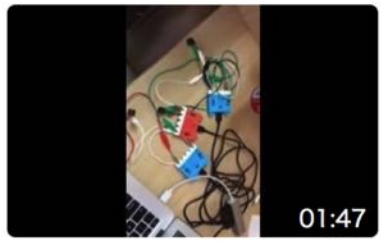
【sessdsa18】第09组：多功能测量仪

收藏于5-7



【sessdsa18】第03组：捣蛋机器人

收藏于5-7



【sessdsa18】第27组 音乐合奏—天空之城

收藏于5-7

【sessdsa18】第16组：paper.io多人对抗游戏

收藏于5-7

【sessdsa18】第25组：双人射击对战游戏

收藏于5-7

【sessdsa18】第04组：中文语音合成播报

收藏于5-7

【sessdsa18】第31组：微钢琴

收藏于5-7

【sessdsa18】第26组：俄罗斯方块

【sessdsa18】第20组：野外探险工具箱

【sessdsa18】第24组：皮坎旋律（音乐游戏）

【sessdsa18】第06组：简易版坦克大战

【sessdsa18】第10组：默契大考验之你们凉凉了么

【sessdsa18】第13组：宇宙飞船系列游戏

收藏于5-7

【sessdsa18】第22组：疯狂炸弹人（超长3分钟）

收藏于5-7

【sessdsa18】第17组：迷宫

收藏于5-7

【sessdsa18】第18组：复古小游戏机

收藏于5-7

【sessdsa18】第28组：Micro:musicbox

收藏于5-7

【sessdsa18】第33组：多功能搬运车

收藏于5-7

【sessdsa18】第11组：音乐游戏MusicBlocks

收藏于5-7

【sessdsa18】第08组：音乐编辑器

收藏于5-7

【sessdsa18】第15组：深蹲计数&猜旋律

收藏于5-7

【sessdsa18】第34组：Microbit 实现超级马里奥

收藏于5-7

【sessdsa18】第30组：无敌乒乓球

【sessdsa18】第02组：模拟电报机

【sessdsa18】第29组：数算课的生死时速

【sessdsa18】第35组：micro:bit模拟器

【sessdsa18】第14组：遥控向日葵花车

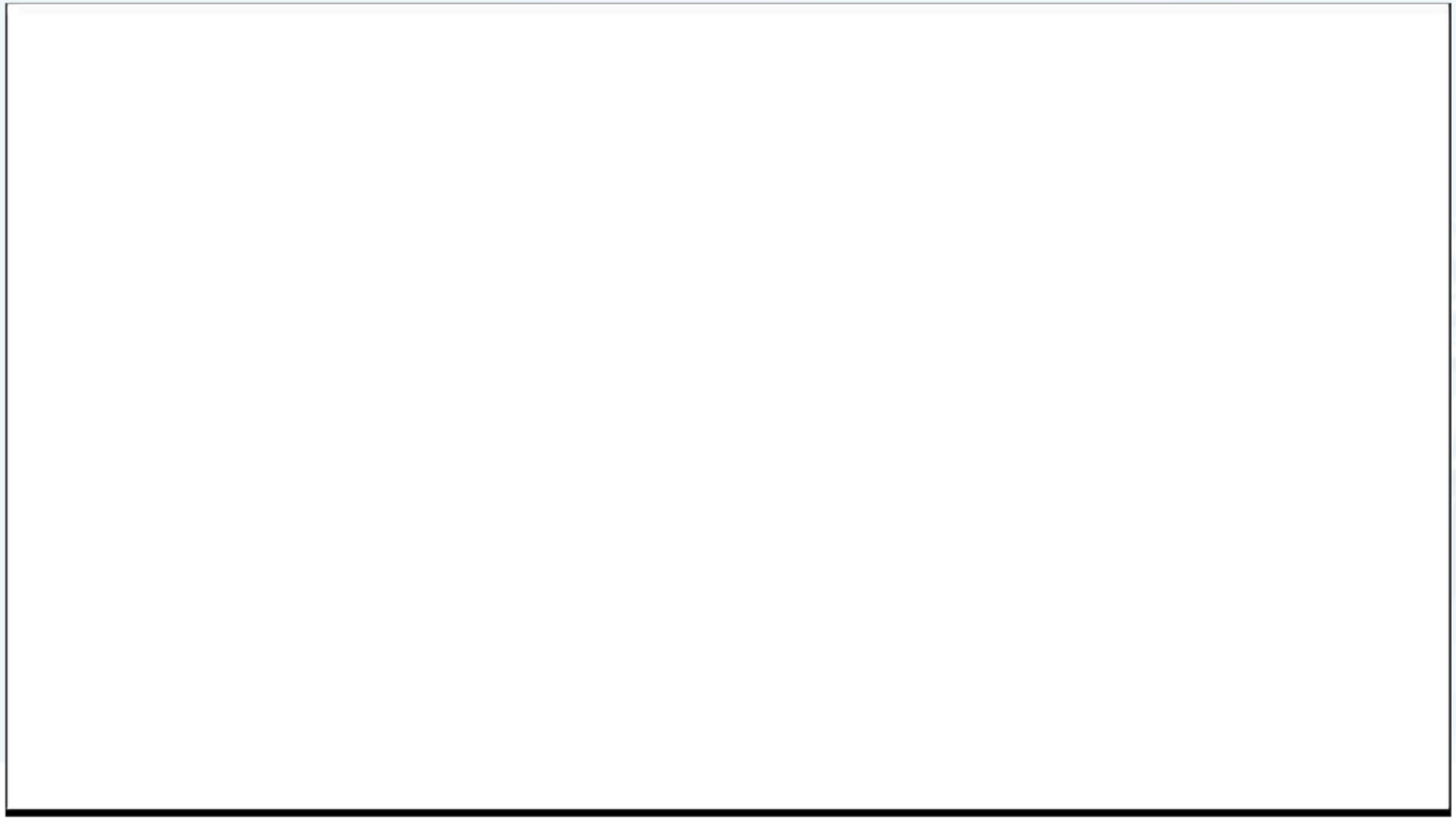
多功能搬运车

microbit模拟器

微钢琴


Micro: bit ×piano/Deemo
微钢琴演奏美妙乐章
Micro:bit实现发明梦想
A:选择曲目
B:确认开始
屏幕显示曲名
琴键下落
or
显示乐名
备选曲目:
Pure White
根据谱子弹奏
或完全自由演奏
弹得不错
or
再接再厉
之后可以开始新的一局游戏

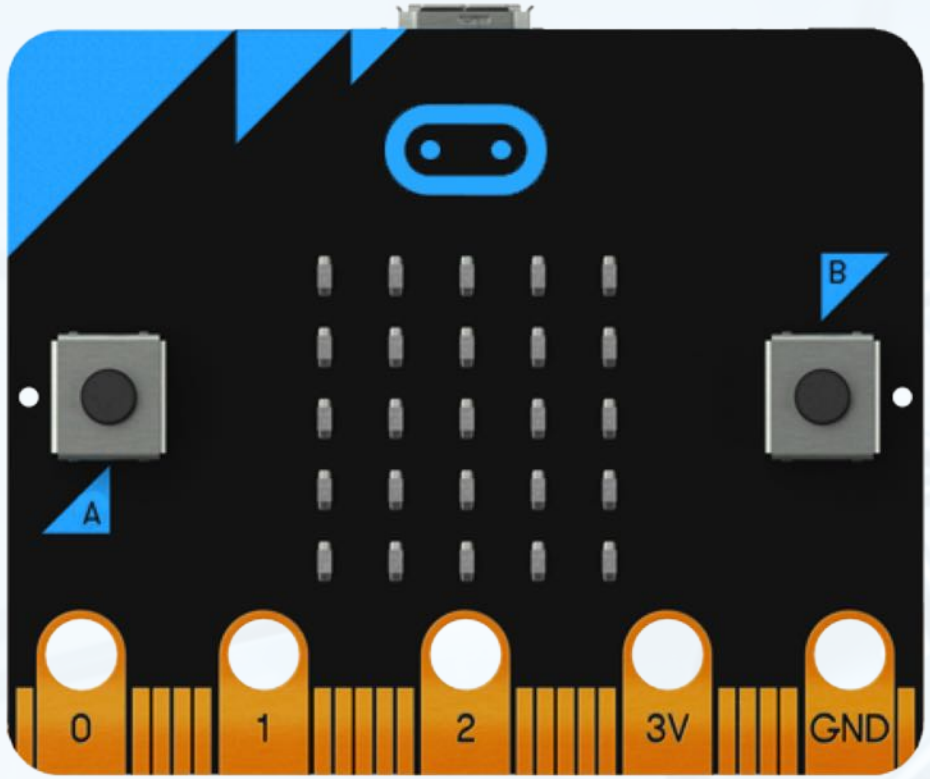
paper.io多人对抗游戏



俄罗斯方块

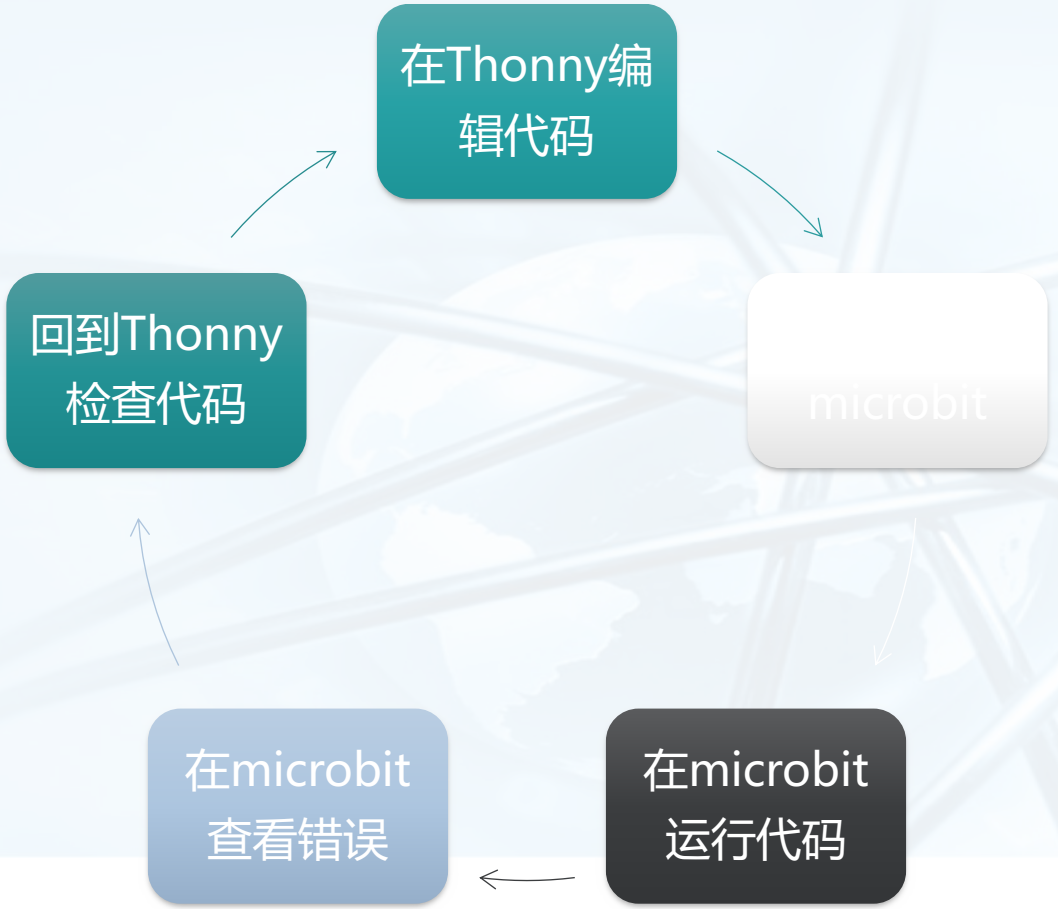
音乐编辑器

目录



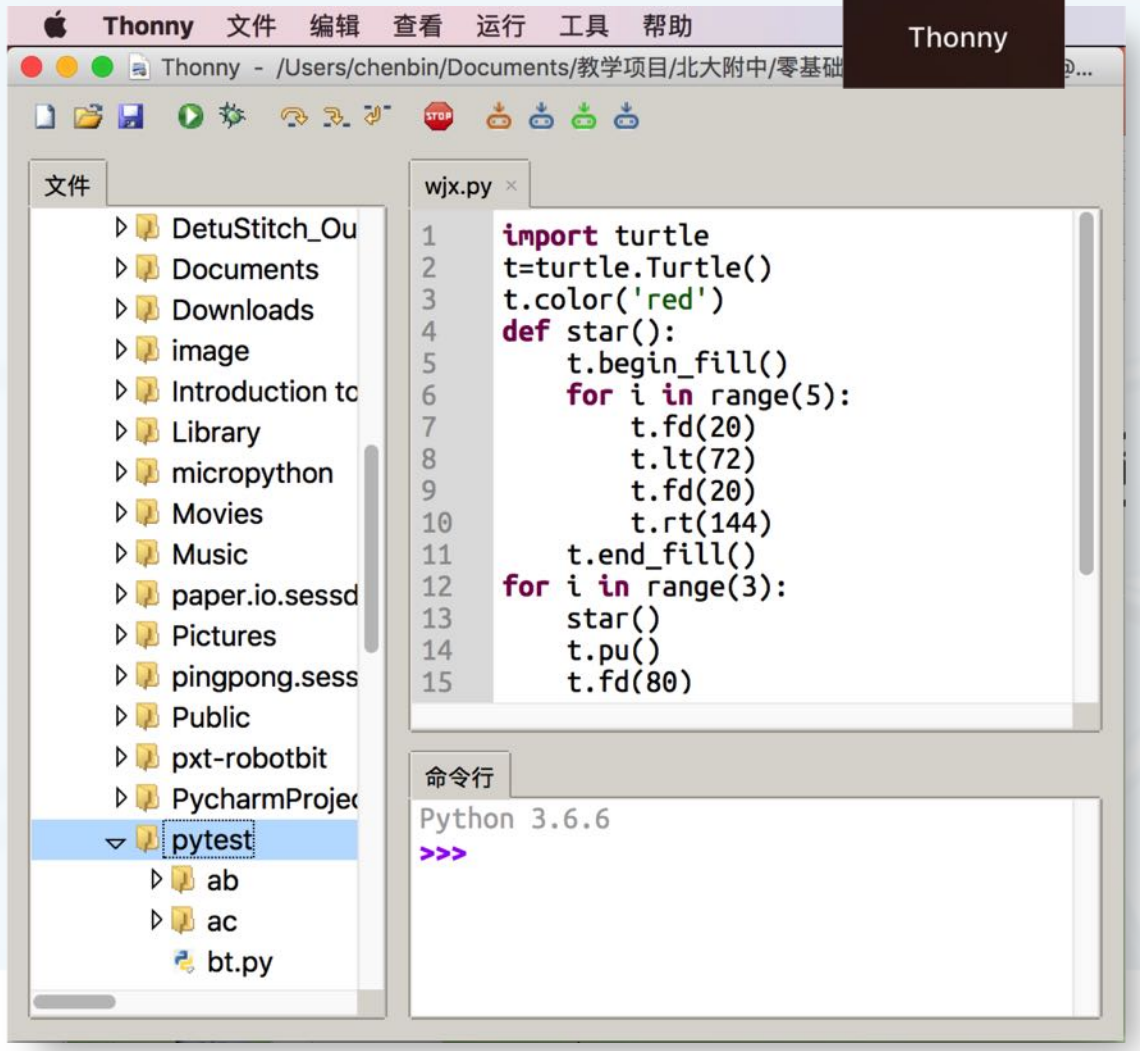
microbit-micropython编程的过程

- 在PC上编写程序
- 下载到microbit运行，并观察运行结果
- microbit作为单片机可以脱离PC自主运行程序，只需要正常供电即可
- 有错误的话再回到PC上修改
- 重复上述过程



集成开发环境：Thonny

- 跨平台Windows/macOS/Linux
- 自带最新版本Python3，无需安装Python
- 体积小巧，功能齐全
- 安装第三方模块很方便
- 可以连接microbit单片机编程
- 地小空开放实验室汉化版本
  https://github.com/chbpku/dxkStickIDE/tree/master/Setup



初识microbit-micropython

1. USB线连接microbit
2. 打开Thonny
3. 输入程序，保存为py文件
4. 点击“写入运行环境”

   稍等一下，闪烁结束

   出现成功Done字样
5. 点击“写入当前代码”

   出现成功Done字样
6. 立刻运行，可按RESET重启

第一个程序：Hello World!

- microbit基本硬件的访问都在模块microbit中
- 通常，首先导入microbit模块的所有对象
- 我们来写第一个helloworld程序

  电源开关打到OFF，模式开关打到WR，连接USB线

hello.py

```
1  from microbit import *
2
3  display.scroll("Hello, World!")
```

第一个程序：Hello World!

- **可能出现错误**
  在LED屏滚动显示错误

  指示错误代码的行号和错误类型

- **名称错误：NameError**
  注意大小写

- **语法错误：SyntaxError**
  注意中文标点等

- **死机？**
  RESET按钮重启

```
1  from microbit import *
2
3  Display.scroll("Hello, World!")
```

第一个程序：Hello World!

- **在写入代码之前发生的错误**
- **常见：IndentationError**

  缩进错误
- **Python语言缩进规则**

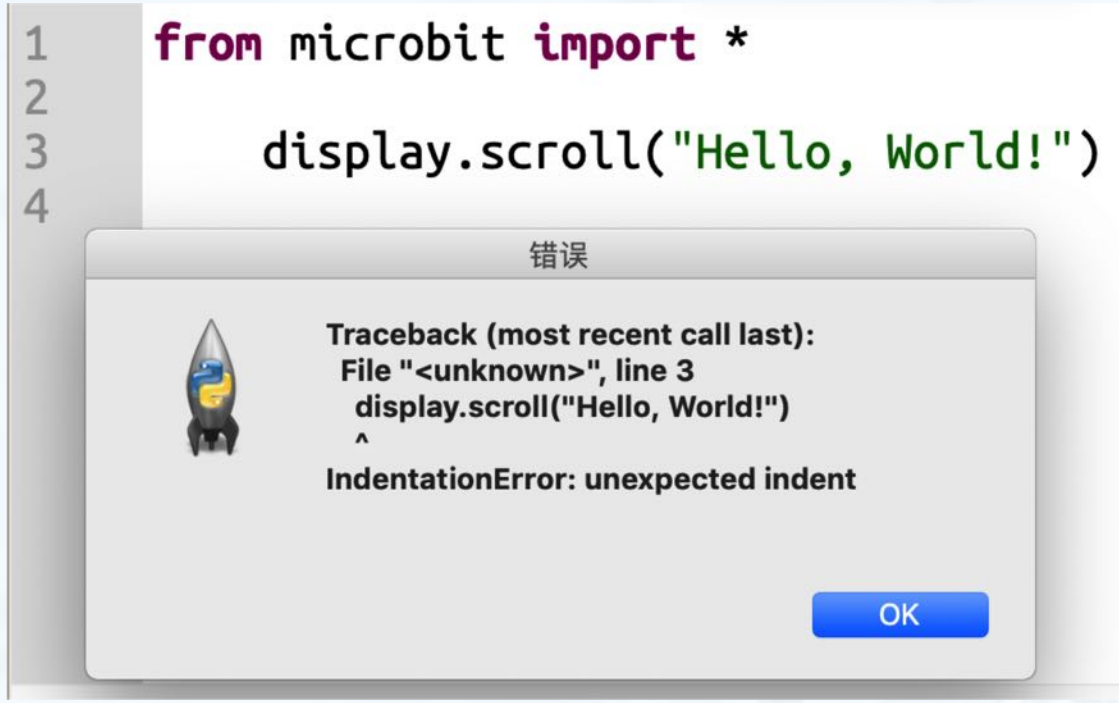
  同一层次一律左对齐

  有冒号结尾的语句包含语句块一律缩进

  - if, elif, else, for, while, def, class, with

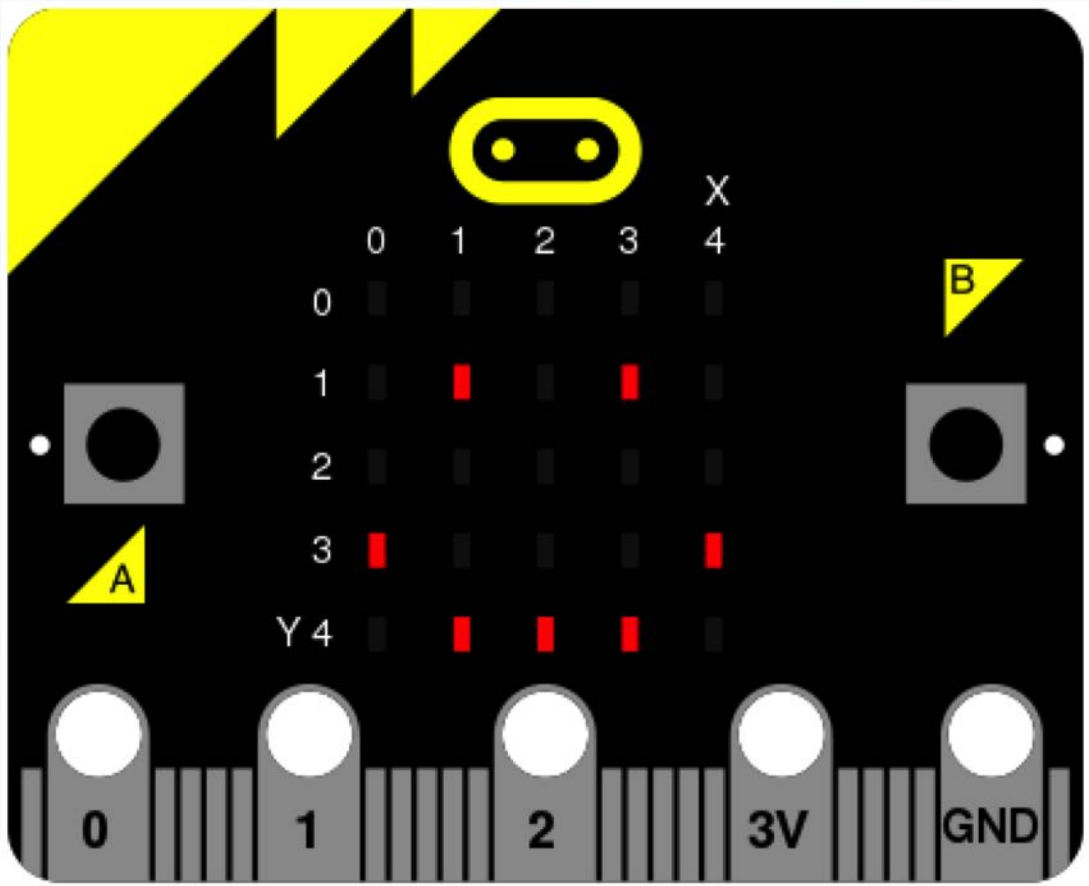
  同一个源代码文件中缩进要一致

  一般是4个空格



图像Image

- 5*5 LED点阵可构成图像显示
  x,y坐标（0,0)~(4,4)
- 每个LED亮度0～9
  0=off
- 用display.show显示Image类对象
  内置Image对象的图像
  自定义的Image对象
  动画：Image对象序列



内置Image对象

- microbit模块内置了数十个Image对象，可以直接调用

  Image.HAPPY

- 分类

  表情类

  时钟类：CLOCK1~12

  方向类：ARROW_N/E/W/S

  形状类：TRIANGLE…

  动物类：RABBIT…

  杂物类：HOUSE/SKULL…

- Image.HEART
- Image.HEART_SMALL
- Image.HAPPY
- Image.SMILE
- Image.SAD
- Image.CONFUSED
- Image.ANGRY
- Image.ASLEEP
- Image.SURPRISED
- Image.SILLY
- Image.FABULOUS
- Image.MEH
- Image.YES
- Image.NO

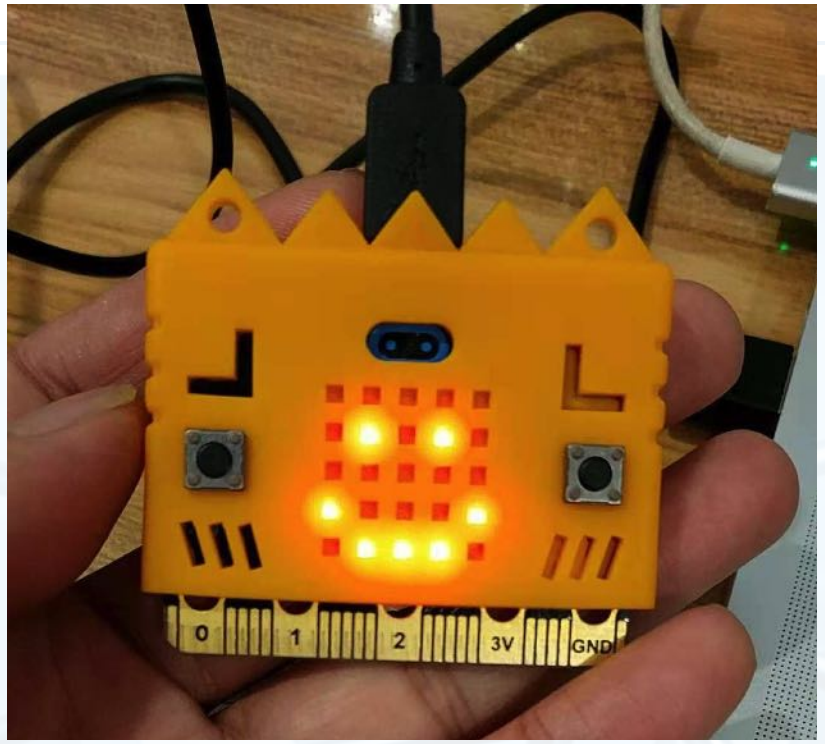


image1.py

```
1  from microbit import *
2
3  display.show(Image.HAPPY)
```

自定义Image图像

- **用Image类来生成**

  `Image(str)`

- **指定格式字符串**

  字符0～9对应LED灯亮度

  :或者\n对应换行

- **Python字符串跨行表示**

  挨在一起的多个引号字符串

  表示一个字符串常量

  `"05050:05050:05050:99999:09990"`

image2.py

```
1  from microbit import *
2
3  boat = Image("05050:"
4               "05050:"
5               "05050:"
6               "99999:"
7               "09990")
8  display.show(boat)
```

显示动画

- **用图像序列来显示动画**

  `display.show(seq,delay=400,wait=True,loop=False,clear=False)`

  可以是列表或者元组，可以是内置图像或自定义图像

  可以指定帧间隔时间delay，是否动画结束再执行下一条语句wait

  是否不停循环动画loop，是否动画结束后清除显示clear

- **内置的两个图像列表**

  `Image.CLOCKS, Image.ARROWS`

image3.py

```
1  from microbit import *
2
3  display.show(Image.ALL_CLOCKS, loop=True, delay=100)
```

动画：沉船

image4.py

```
from microbit import *

boat1 = Image("05050:"
              "05050:"
              "05050:"
              "99999:"
              "09990")
boat2 = Image("00000:"
              "05050:"
              "05050:"
              "05050:"
              "99999")
boat3 = Image("00000:"
              "00000:"
              "05050:"
              "05050:"
              "05050")
boat4 = Image("00000:"
              "00000:"
              "00000:"
              "05050:"
              "05050")
boat5 = Image("00000:"
              "00000:"
              "00000:"
              "00000:"
              "05050")
boat6 = Image("00000:"
              "00000:"
              "00000:"
              "00000:"
              "00000")
all_boats = [boat1, boat2, boat3, boat4, boat5, boat6]
display.show(all_boats, delay=200)
```

LED点阵屏控制display

- **开关显示屏**
  `display.on(), off()`
- **显示字符串或者图像**
  `display.show(s)`
- **滚动显示字符串**
  `display.scroll(s)`
- **清除显示**
  `display.clear()`
- **点亮一个像素(b=0~9)**
  `display.set_pixel(x,y,b)`

image5.py

```
1   from microbit import *
2
3   display.show('HELLO')
4   sleep(1000)
5   display.scroll('WORLD')
6   sleep(1000)
7   display.clear()
8   for i in range(5):
9       display.set_pixel(i, i, 9)
10      sleep(200)
11  display.clear()
12  display.scroll('BYE!')
13
```

按钮控制button_a / button_b

- **有两个对象和三个方法**

  `button_a, button_b`

- **一直按着按钮**

  `is_pressed()`

- **按下-放开按钮**

  `was_pressed()`

  调用将清除状态

- **按过按钮的次数（距上次调用）**

  `get_presses()`

button1.py

```
1  from microbit import *
2
3  display.show(Image.ALL_CLOCKS, wait=True)
4  display.scroll(str(button_a.get_presses()))
```

条件循环while

- **检测条件是否成立**

  成立则执行语句块

  执行完毕则再次检测条件

  直到条件不成立，执行else

  while语句结束

- **系统函数running_time**

  返回启动开始的毫秒数

  running_time()

- **microbit没有时钟硬件模块，掉电就丢失时间**

button2.py

```
1  from microbit import *
2
3  while running_time() < 5000:
4      display.show(Image.ASLEEP)
5  else:
6      display.show(Image.SURPRISED)
```

事件循环和处理

- 如果是检测按钮动作，一般需要无限循环来等待事件发生

  while True:

  判断is_pressed()是否True

- 可以用逻辑运算符连接条件

  同时成立and

  任一成立or

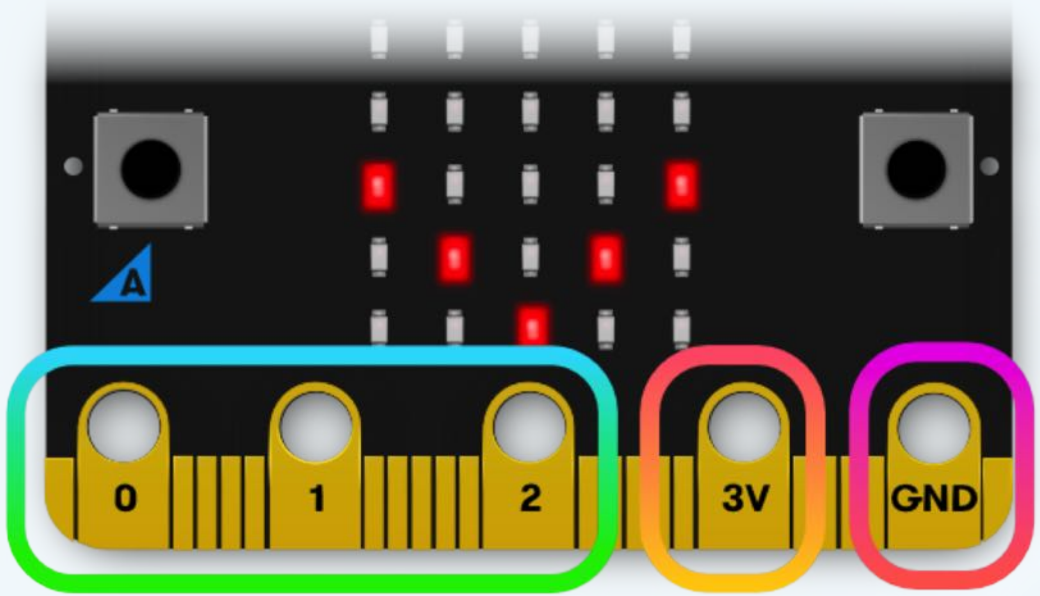
  不成立not

- 两个按钮同时按下？

button3.py

```
1   from microbit import *
2
3   while True:
4       if button_a.is_pressed():
5           display.show(Image.HAPPY)
6       elif button_b.is_pressed():
7           break
8       else:
9           display.show(Image.SAD)
10  display.clear()
```

输入/输出引脚：pin0~pin20



- **microbit模块中的内置对象**

  `pin0~pin16; pin19, pin20`

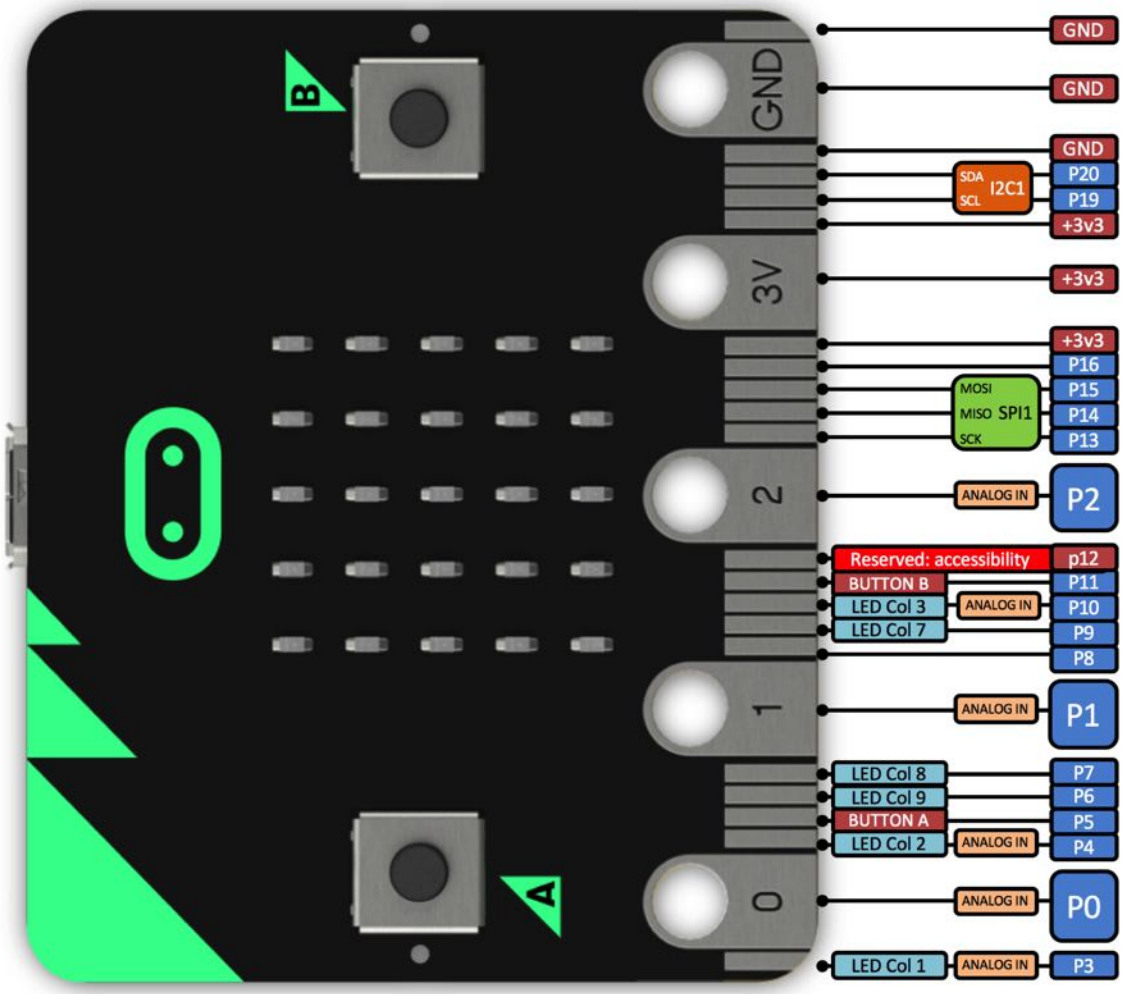
- **分为3个不同的类**

  `MicroBitTouchPin: pin0~2`

  `MicroBitAnalogDigitalPin: pin3,4,10`

  `MicroBitDigitalPin: pin5~9,pin11~16,pin19~20`

输入/输出引脚：三种类



序号	名称	类型	作用
1	P0	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
2	P1	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
3	P2	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
4	P3	LED Col1/ADC-GPIO	如果关闭LED，可以当作ADC-GPIO用
5	P4	LED Col2/ADC-GPIO	如果关闭LED，可以当作ADC-GPIO用
6	P5	Button-A	按钮A
7	P6	LED Col9/D-GPIO	如果关闭LED，可以当作D-GPIO用
8	P7	LED Col8/D-GPIO	如果关闭LED，可以当作D-GPIO用
9	P8	D-GPIO	空闲的数字输入输出D-GPIO
10	P9	LED Col7/D-GPIO	如果关闭LED，可以当作D-GPIO用
11	P10	LED Col3/D-GPIO	如果关闭LED，可以当作ADC-GPIO用
12	P11	Button-B	按钮B
13	P12	D-GPIO	空闲的数字输入输出D-GPIO
14	P13	D-GPIO/SPI-SCK	缺省的SPI-SCK信号
15	P14	D-GPIO/SPI-MISO	缺省的SPI-MISO信号
16	P15	D-GPIO/SPI-MOSI	缺省的SPI-MOSI信号
17	P16	D-GPIO/SPI-NSS(CS)	空闲的数字输入输出D-GPIO，可用作SPI芯片选择
18	P17	3v3	电源输入
19	P18	3v3	电源输入
20	P19	I2C-SCL	I2C总线的时钟信号（内置加速计/指南针）
21	P20	I2C-SDA	I2C总线的数据线（内置加速计/指南针）
22	P21	GND	接地
23	P22	GND	接地

引脚输入：MicroBitTouchPin类

- **大引脚pin0~2的触摸操作**

  一手接触GND

  一手接触P0～2

  从pin0.is_touched()读出

- **触摸操作会有诸多因素影响，可能会不触发事件**

- **通常可以给一个延迟时间**

  sleep(50)

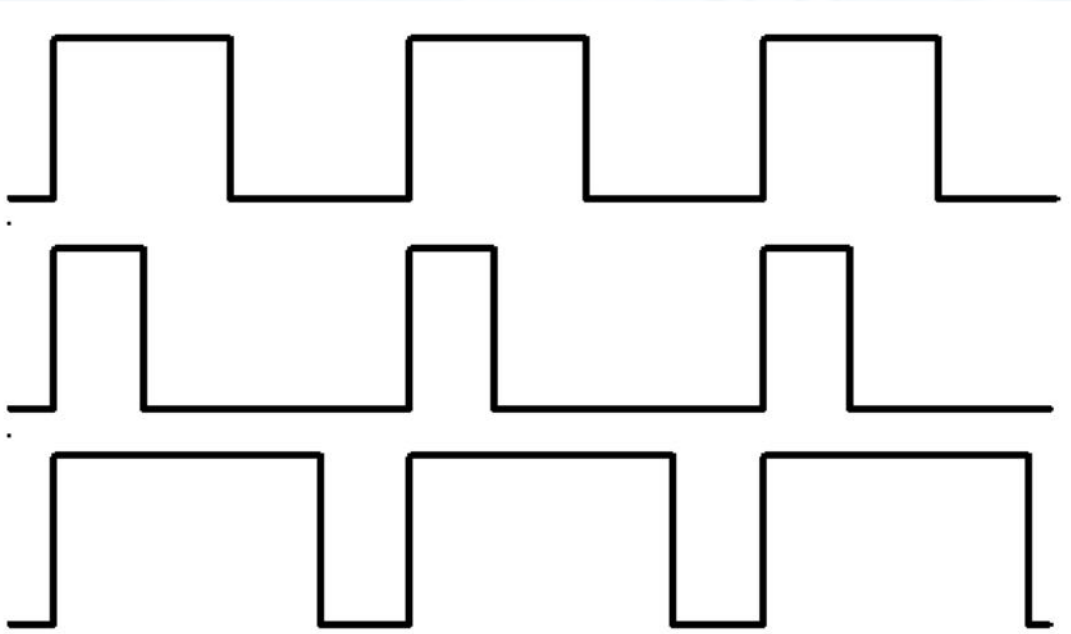
pin1.py

```
1  from microbit import *
2
3  while True:
4      if pin0.is_touched():
5          display.show(Image.HAPPY)
6      else:
7          display.show(Image.SAD)
8      sleep(50)
```

引脚输入输出：模拟数字转换

- **数字信号Digital**
  0V等于0；3.3V等于1
- **模拟电压信号Analog**
  0V～3.3V之间
- **模拟转数字（采样）ADC**
  MicroBitAnalogDigitalPin
  MicroBitTouchPin
  1024级：0V=0，3.3V=1023
- **数字转模拟（PWM）**
  所有的pin都可以0～1023输出

write_analog(511)
write_analog(255)
write_analog(767)



脉冲转换为声音（套件P0接了扬声器）

pin2.py

```
1    from microbit import *
2
3    while True:
4        pin0.write_digital(1)
5        sleep(20)
6        pin0.write_digital(0)
7        sleep(480)
```

pin3.py

```
1    from microbit import *
2
3    pin0.write_analog(511)
```

音乐和语音合成

- **music模块可以从引脚输出音乐，由喇叭播放**
  - 内置音乐乐曲
  - 由音符编写乐曲
  - 发出指定频率声音
- **speech模块输出语音合成**
  - say单词
  - pronounce音节
  - sing用指定频率说音节
  - 试验性的模块

music1.py

```
1  import music
2
3  music.play(music.BIRTHDAY, wait=False)
```

内置乐曲

- microbit内置了一些乐曲

```
DADADADUM, ENTERTAINER
PRELUDE, ODE, NYAN
RINGTONE, FUNK, BLUES
BIRTHDAY, WEDDING
FUNERAL, PUNCHLINE
PYTHON, BADDY, CHASE
BA_DING, WAWAWAWAA
JUMP_UP, JUMP_DOWN
POWER_UP, POWER_DOWN
```

音符和组成乐曲

音符的格式

音符[八度][:时长]

音符：CDEFGAB，#，b，R

八度：0～8，4是中音

时长：整数，tick的数量

music.play

单个音符，或者音符的序列

pin=pin0：播放的引脚

wait=True：等待播放结束

loop=False：无限循环

music2.py

```
1  import music
2
3  tune = ["C4:4", "D4:4", "E4:4", "C4:4",
4          "C4:4", "D4:4", "E4:4", "C4:4",
5          "E4:4", "F4:4", "G4:8",
6          "E4:4", "F4:4", "G4:8"]
7  music.play(tune)
```

指定频率发声

- **music.pitch()**

  指定频率（Hz）

  duration=-1，无限或指定毫秒

  pin=pin0

  wait=True

- **music.stop()**
- **music.reset()**

music3.py

```
1  import music
2
3  while True:
4      for freq in range(880, 1760, 16):
5          music.pitch(freq, 6)
6      for freq in range(1760, 880, -16):
7          music.pitch(freq, 6)
```

综合练习：反应力小游戏

- 在指定时间内需要同时按下AB键可以升级
  时间从1000ms开始每级减100ms
- 显示级别（1~10）
- 显示Image.ASLEEP/音乐music.POWER_UP
- 显示3、2、1、Image.TARGET
- 等待指定时间
- 成功按钮则显示Image.HAPPY/音乐music.JUMP_UP
  升级，减时间，10级祝贺音乐，退出游戏
- 失败则显示Image.SKULL/音乐music.WAWAWAWAA
  回到显示级别，重来

随机数发生器

- **random模块是伪随机数发生器**

  seed(整数)

  二进制整数getrandbits(位数)

  randint(start, end)

  randrange(start, end)

  浮点数：random()

  浮点数：uniform(start, end)

  序列选择：choice(序列)

- **伪随机数pseudo random**

  根据算法生成的数值序列，均匀分布，每个seed对应一个序列

修改代码为按钮显示随机图像

random1.py

```
1  from microbit import *
2  import random
3
4  random.seed(1337)
5  while True:
6      if button_a.was_pressed():
7          display.show(str(random.randint(1, 6)))
8          sleep(500)
9          display.clear()
```

移动检测：加速计accelerometer

- **检测3DoF（三自由度）的移动**
  绕x/y/z轴旋转运动

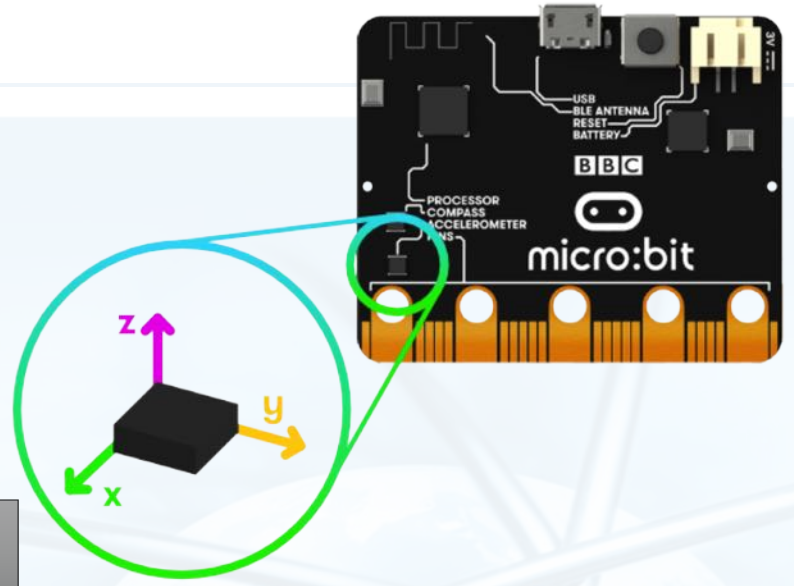
- **基本方法**
  每个自由度返回正负值

  `get_x()`

  `get_y()`

  `get_z()`

  `get_values()`



acce1.py

```
1   from microbit import *
2
3   while True:
4       reading = accelerometer.get_x()
5       if reading > 20:
6           display.show("R")
7       elif reading < -20:
8           display.show("L")
9       else:
10          display.show("-")
```

魔音盒：控制发声频率

- 倾斜的角度对应发声频率

```
accelerometer.get_y()
```

```
music.pitch(freq,10)
```

- 修改为控制一个像素点？

```
accelerometer.get_x()
```

```
accelerometer.get_y()
```

```
display.set_pixel(x,y,b)
```

acce2.py

```
1  from microbit import *
2  import music
3
4  while True:
5      music.pitch(accelerometer.get_y(), 10)
```

姿态识别gesture

- **综合运动的姿态识别**

  current_gesture()

  was_gesture()

- **常用的姿态**

  上up/下down/左left/右right

  面朝上face up/面朝下face down

  自由落体freefall

  加速度运动3g/6g/8g

  摇shake

写一个沙漏　程序

acce3.py

```
1  from microbit import *
2  import random
3
4  while True:
5      display.show("*")
6      if accelerometer.was_gesture("shake"):
7          display.show(str(random.randint(1, 6)))
8          sleep(1000)
```

指南针

- **内置的指南针传感器compass**

  is_calibrated(): 是否校准

  calibrate(): 进行校准

  clear_calibrate(): 清除

  heading():方向0～359度

- **指北针**

  利用Image.ALL_CLOCKS序列

  1～12来指示正北

compass1.py

```
1  from microbit import *
2
3  compass.calibrate()
4
5  while True:
6      sleep(100)
7      needle = ((15 - compass.heading()) // 30) % 12
8      display.show(Image.ALL_CLOCKS[needle])
```

指南针：金属探测器

- **磁场强度**

  `get_field_strength()`

- **距离金属越近磁场强度越大**

compass2.py

```
1  from microbit import *
2  import music
3
4  # 金属探测器
5  while True:
6      fs = compass.get_field_strength()
7      if fs > 60000:
8          music.pitch(fs // 100, 10)
9      sleep(200)
```

彩灯NeoPixel模块

写一个秒针效果的程序

- 创建neopixel对象

```
from microbit import *
import neopixel
np = neopixel.NeoPixel(pin16,12)
```

- 设置彩灯（0~11）

```
np[0]=(255,0,0)
np[-1]=(0,255,0)
```

- 点亮/清除彩灯

```
np.show()
np.clear()
```

np1.py

```
1   from microbit import *
2   import neopixel
3
4   np = neopixel.NeoPixel(pin16, 12)
5   i = 0
6   while True:
7       np.clear()
8       np[i] = (30, 0, 0)
9       np[(i + 6) % 12] = (0, 30, 0)
10      np.show()
11      i = (i + 1) % 12
12      sleep(200)
```

无线通信radio

通过频道发送和接收消息

config(channel=7)：设置

on()：打开无线

send()：发送字符串

receive()：接收字符串

off()：关闭无线

关于频道

缺省值为7

可以是0～83之间的一个频道

不同频道之间无法通信

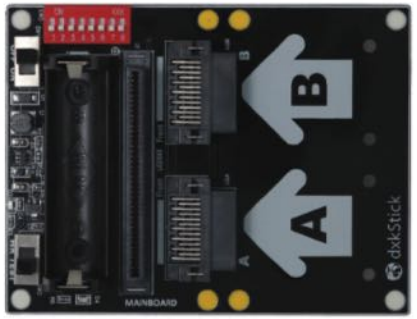
radio1.py

```
from microbit import *
import radio
import neopixel

np = neopixel.NeoPixel(pin16, 12)

# 自己的颜色
my = "60,0,0"

# 记录收到的颜色队列
clst = [(0, 0, 0) for i in range(12)]
radio.on()
while True:
    # 按钮控制自己发送颜色
    if button_a.was_pressed():
        radio.send(my)
    # 接收空中的颜色
    r = radio.receive()
    if r is not None:
        display.show(Image.YES)
        clst.pop(0)
        clst.append(tuple(map(int, r.split(","))))
    else:
        display.show(Image.ASLEEP)
    # 显示颜色队列
    np.clear()
    for i in range(12):
        np[i] = clst[i]
    np.show()
    # 停留一会儿
    sleep(100)
```

程序汇总和教学要点

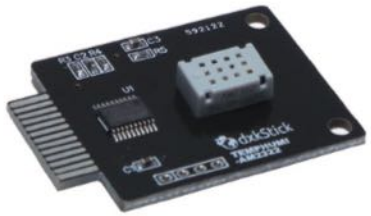
- **第一行程序hello.py**
  观察和修正错误
- **图像image1~5.py**
  对象和赋值
  字符串表示
  序列和列表的应用
  函数的关键字参数
- **按钮button1~3.py**
  嵌套的函数调用
  条件循环while
  事件循环和处理
- **输入输出引脚pin1~3.py**
  采样和PWM
- **音乐、语音合成music1~3.py**
  迭代循环for
- **随机数发生器random1.py**
  从序列中随机选择元素
  伪随机数
- **移动和姿态识别acce1~3.py**
- **指南针compass1~2.py**
- **彩灯np1.py**
  取余数的应用

dxkStick Python语言教学套件



DSK000
MAINBOARD
主板
x1

DSK001
DISPLAY-OLED12864
12864OLED显示屏
x1



DSK002
TEMPHUMI-AM2322
温湿度传感器
x1

DSK003
LED-RED
单头LED红色
x1

DSK004
POTEN
模拟电位器
x1

DSK005
LIGHT
光敏电阻
x1

DSK006
SOUND
MIC声音传感器 x1

DSK007
GAMEPAD
游戏手柄 x1

DSK008
ULTRASONIC-SR04
超声波测距 x1

DSK900
MICROBIT
microbit单片机 x1

DSK901
ADAPTOR-CARD-RJ11
转接卡RJ11 x1

DSK902
ADAPTOR-SLOT-RJ11
转接槽RJ11 x1

DSK903
WIRE-20CM-RJ11
20厘米RJ11六芯线 x1

DSK904
WIRE-MICROUSB
microUSB线 x1

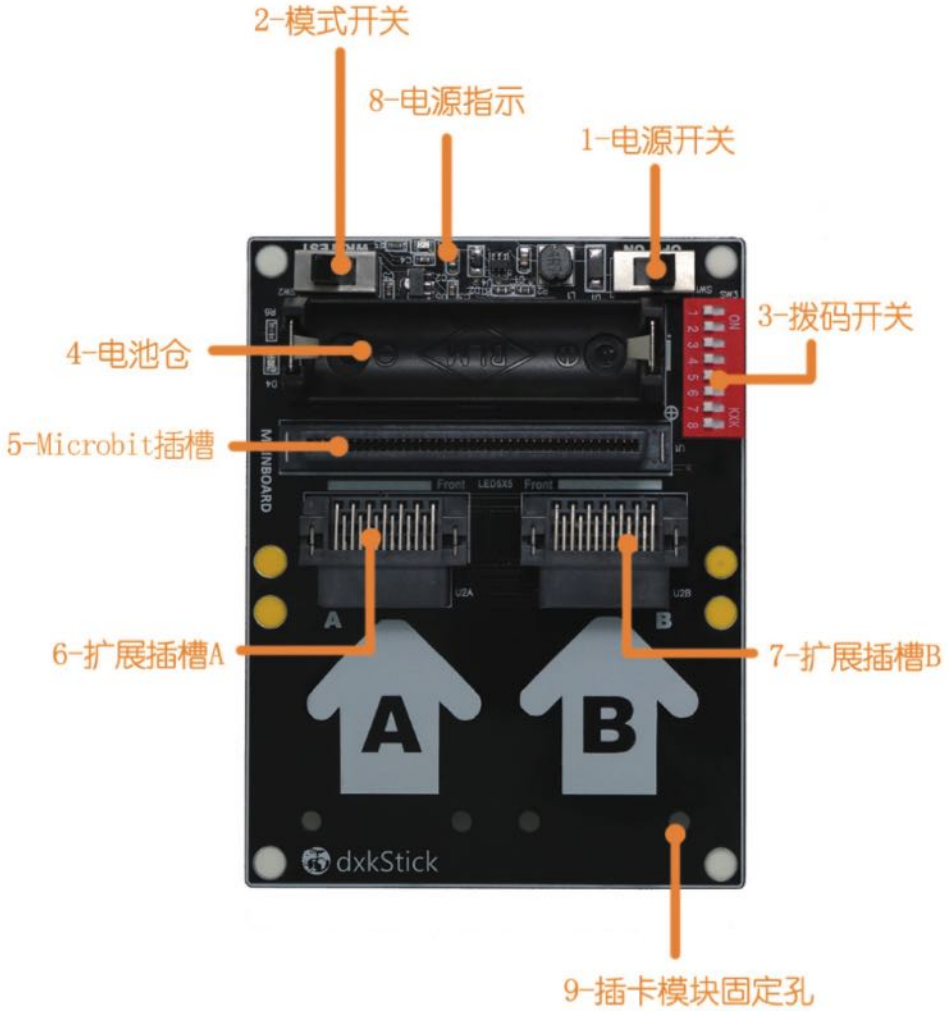
DSK905
SI-CASE
硅胶保护套 x1

DSK906
HEX-SCREWDRV
内六角螺丝刀 x1

DSK907
SCREW-BOX
螺丝盒子 x1

3.0 连接和开启

3.1 主板安装设置



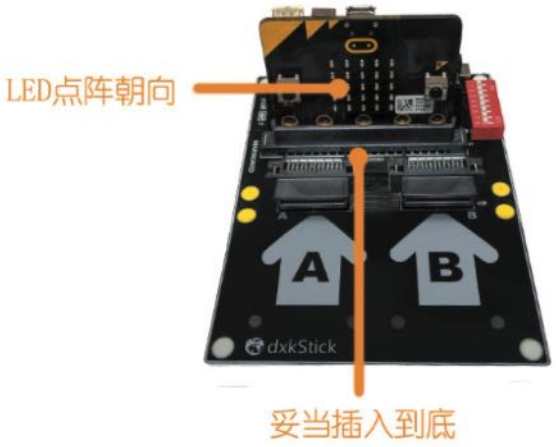
序号-名称	功能
1-电源开关	开启ON-关闭OFF电源
2-模式开关	切换工作WR-测试TEST模式
3-拨码开关	设置（1-5）无线通信频道CH；（6-8）同频道分组GRP 以二进制进行编码，ON的方向为1，小数字为低位。
4-电池仓	安装5号电池，注意正负极标志
5-Microbit插槽	安装microbit单片机，标注LED5X5为点阵的朝向
6-扩展插槽A	可以插入各型号插卡模块
7-扩展插槽B	可以插入各型号插卡模块
8-电源指示	电源开启后会持续发 光
9-插卡模块固定孔	可用螺丝，将插卡模块强力固定在主板上

电池安装供电

本套件主板可以由各种类型的5号电池供电，包括：碳性电池、碱性电池、镍氢电池、锂电池、磷酸铁锂电池。从安全和经济的角度，推荐使用磷酸铁锂电池。

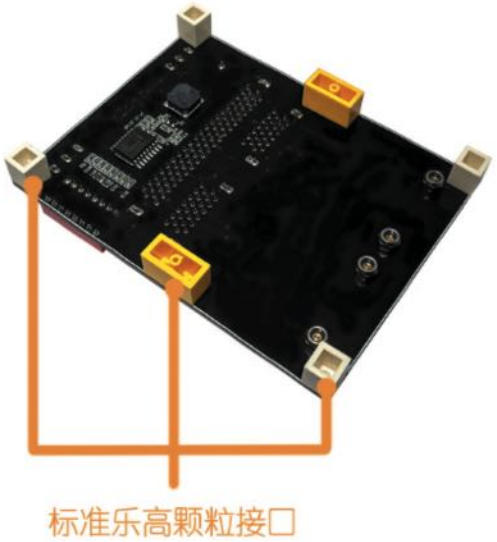
电池在安装到电池仓时，需注意正负极标志，不能反接，并注意电量充足，否则主板将无法正常工作。

Microbit单片机安装



本套件主板的microbit插槽有方向性，但由于microbit单片机没有防呆设计，所以需要辨认单片机上LED点阵的朝向，小心地将microbit单片机妥当插入到插槽底部。

固定到乐高模型



本套件主板的四角和长边中央都已经预装了乐高标准颗粒，用户可以方便地将主板固定到各种乐高拼插积木搭成的模型上，增加学习乐趣，提高套件创造性。

3.2 插卡模块安装

插卡模块安装和固定

本套件主板上有两个插卡插槽，分别标注为A和B，插卡模块可以插入任何一个插槽中。在每个插槽的旁边都有两个固定铜柱，在需要特别加固的情况下，可以用套件附带的内六角螺丝将插卡模块牢牢固定在主板上。

插卡模块的延伸

某些插卡模块的物理尺寸较大，无法插入主板上的插卡插槽，或者需要将模块安装到主板之外的空间时，可以使用模块延伸配件。模块延伸配件包括一个插在主板上的RJ11转接卡，一段RJ11接头的螺旋线，以及一个带有插卡插槽的RJ11转接槽。

由于采用螺旋线延伸插卡模块增加了插卡脱落的风险，建议用内六角螺丝将RJ11转接卡固定在主板上。

固定到乐高模型

本套件主板的四角和长边中央都已经预装了乐高标准颗粒，用户可以方便地将主板固定到各种乐高拼插积木搭成的模型上，增加学习乐趣，提高套件创造性。

3.3 无线联网设置

本套件支持microbit的radio无线编程，并通过自主开发的套件python模块，将主板和各插卡模块的功能融合到microbit无线联网的功能，让初学者能够不需要了解复杂的网络通信知识，就能实现远程通信编程，极大丰富了编程教学的案例和套件的适用范围。

无线联网编程模式需要用到多个dxkStick套件，其中一个套件作为主控节点，其余套件作为受控节点，由主控节点上的程序调用受控节点，获取其连接的传感器信息，或在其连接的显示屏上显示信息。

相同无线频道的节点才可以互相通信。套件主板上的8位拨码开关用于设置无线通信的参数。实际使用中，用户可以将1-5位设置为无线通信的频道，频道号按照二进制设定的整数范围为0—31，在同一个教室中的学生最多可以分为32组做实验而不会互相干扰。

3.3 无线联网设置

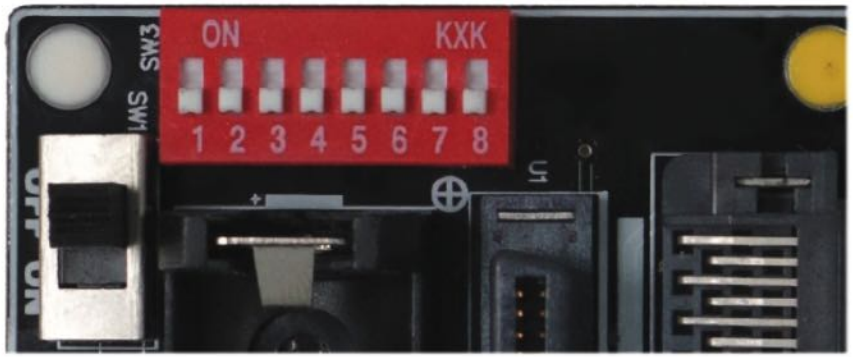
同一频道的几个套件还可以进一步分为几个组，主控节点可以单独控制某个组的受控节点。在一般使用中，可以将每个受控节点分成不同的组，这样就能够单独读取某一个受控节点的传感器数值了，比如，分别读取放置在窗外的温度节点，和放置在门口的温度节点。

拨码开关编号小的为二进制的低位，拨到ON表示1，否则是0。其中1—5位是频道号，6—8位是组号。所以如下的拨码设置为二进制“10010011”，表示频道号19，和组号4。

频道号：19　　组号：4

5	4	3	2	1
1	0	0	1	1

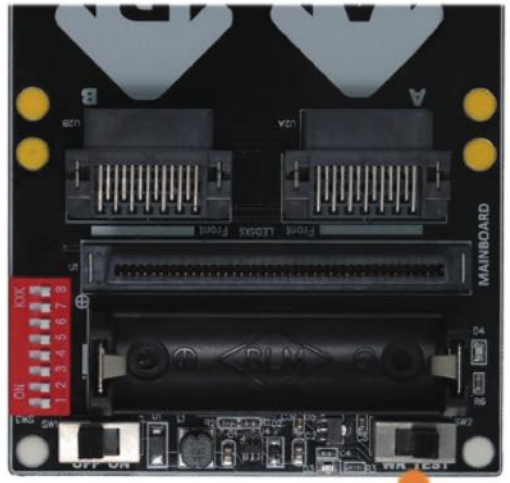
8	7	6
1	0	0

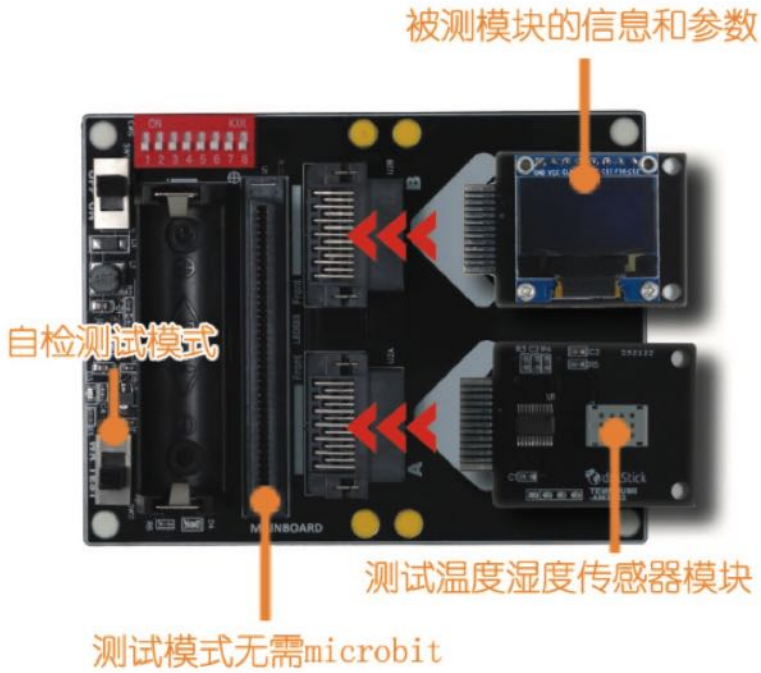


用户设置完拨码开关，即可按照设置的组号进行通信，频道号对用户程序是透明的，无需记录和写入程序，套件底层系统会自动读取频道号来初始化无线通信子系统。

4.0 模块自检

4.1 关于自检模式





自检模式是本套件的特色，用户无需编程就可以检查套件各组件工作是否正常。要启用自检模式，无需安装microbit单片机，但需要将主板上的模式选择开关拨到“TEST”。

OLED显示屏上显示的自检信息分为4行，分别显示套件名称、插卡模块的ID和类型type、模块名称和相应的参数。

自检步骤如下：

1. 将电源开关置于OFF；
2. 安装好电池；
3. 将模式开关置于TEST；
4. 将OLED显示屏模块插入任意一个插槽；
5. 将需要测试的插卡模块插入另一个插槽；
6. 打开电源开关；
7. 观察OLED显示屏上显示的信息。

通过自检测试模式，用户可以确定模块是否完好，同时，也可以抄录插卡模块的ID，用于高级开发编程。

4.2 各模块自检信息

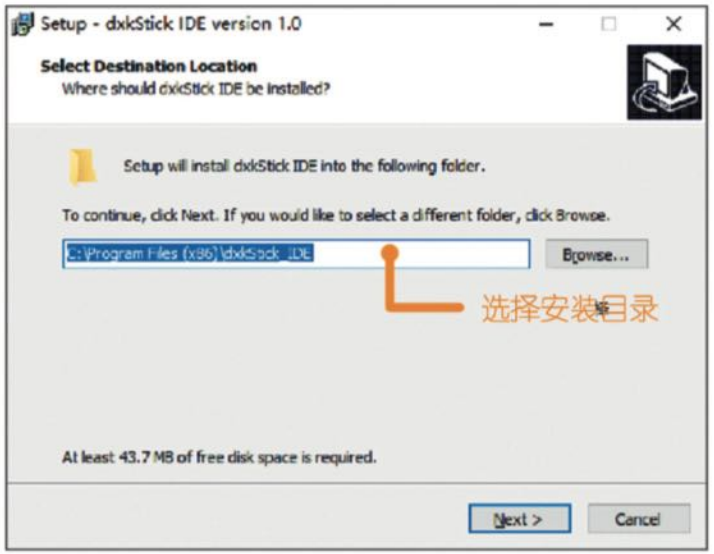
不同的插卡模块会显示不同的自检测试信息，列表说明如下：

代码	模块名称	自检测试信息
DSK001	128640LED 显示屏	（不接其它模块即为自检测试） 显示分辨率，提示内置中文字库
DSK002	温湿度传感器	显示当前的温度值和湿度值
DSK003	单头 LED 红色	显示 ON/OFF，同时闪烁
DSK004	模拟电位器	显示电位器采样值（0-4095），可以旋转旋钮测试
DSK005	光敏电阻	显示当前电阻值，光照越强，电阻值越小
DSK006	MIC 声音传感器	显示当前检测的声音强度
DSK007	游戏手柄	显示每个按钮按下的情况，以及转轴的采样值
DSK008	超声波测距	显示最近障碍的距离（单位 cm）

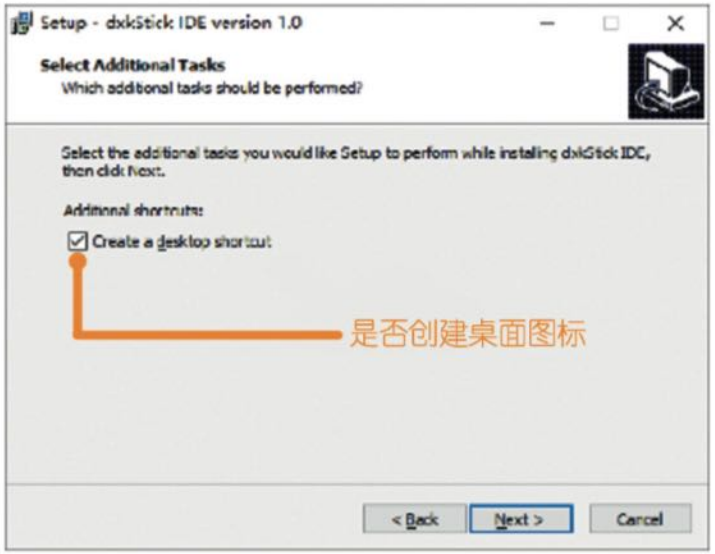
提示：游戏手柄和超声波测距模块可以用模块延伸配件延长测试。

6.0 Python教学编程环境

6.1 安装和启动编程环境



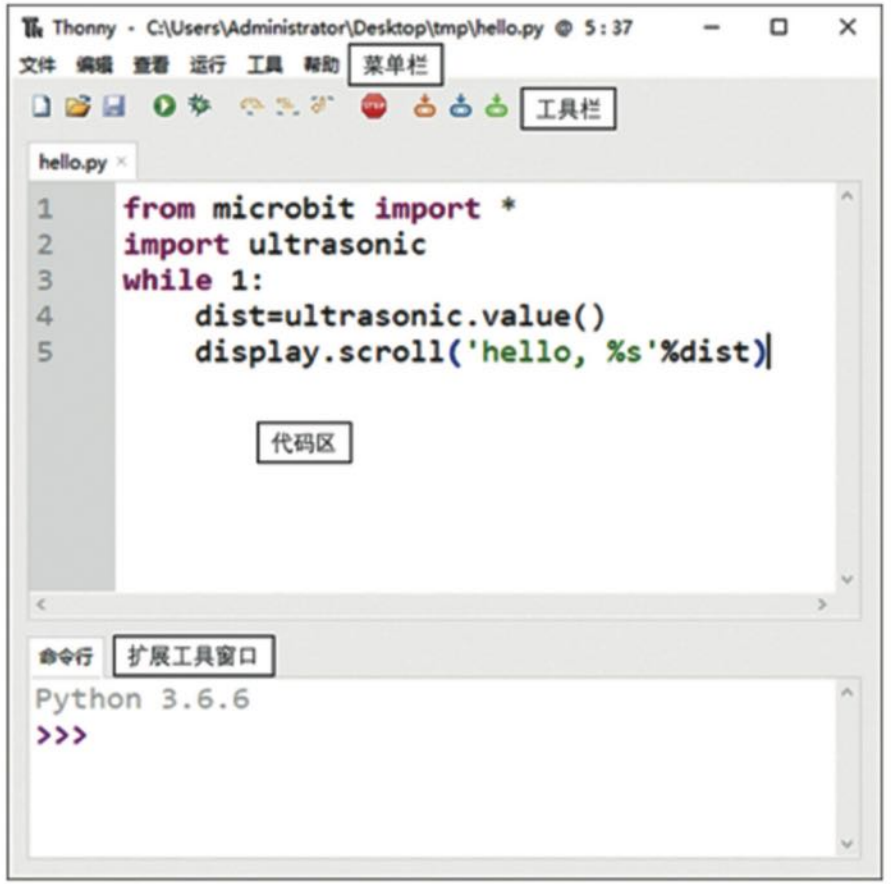
a. 运行安装程序，选择安装目录，并点击“Next”



b. 勾选该条目可以在桌面创建图标

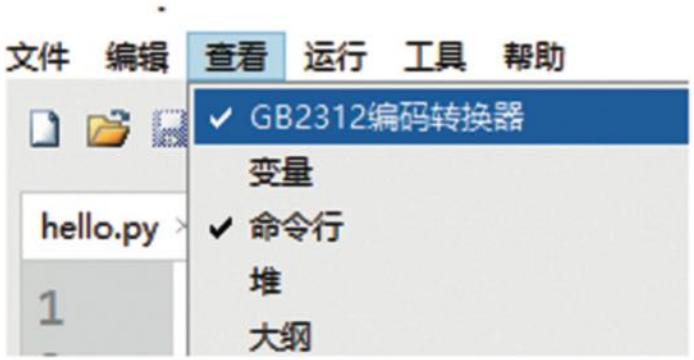
c. 安装完成后桌面与开始菜单中将出现“dxkStick IDE”图标

6.2 界面功能简介

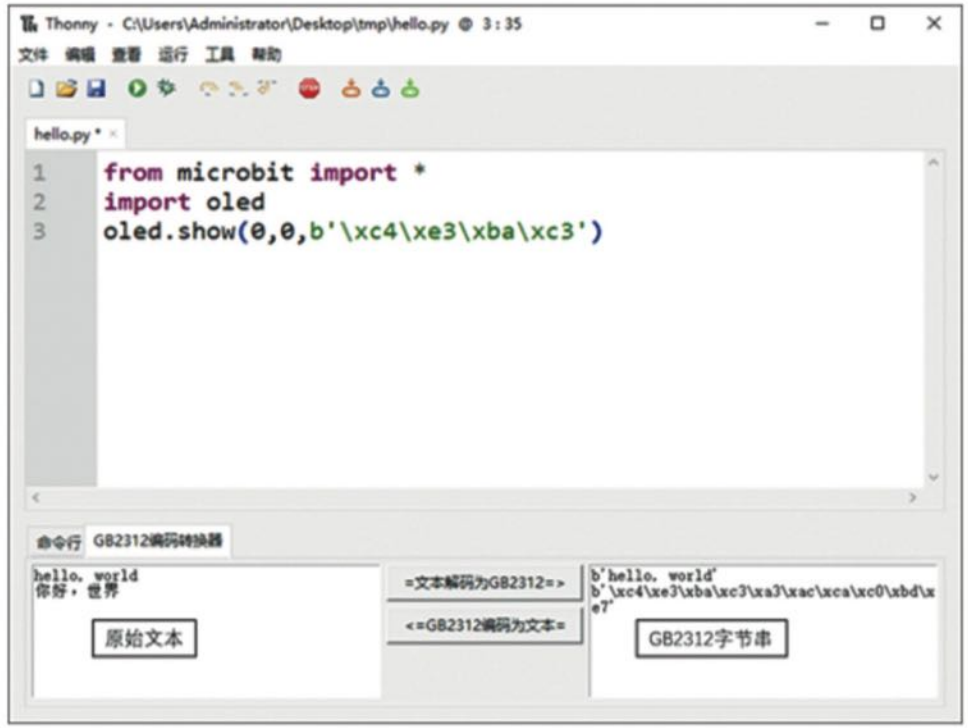


6.2.1 中文编码转换器

由于套件中的OLED显示屏只支持GB2312格式的中文编码，编写程序时需要将需要显示的中文字符转换为GB2312格式；在开发环境中提供了内置的GB2312编码转换工具。



a. 在“查看”选项卡下选择“GB2312编码转换器”使其打勾



b. 窗口下部将显示“GB2312编码转换器窗口”

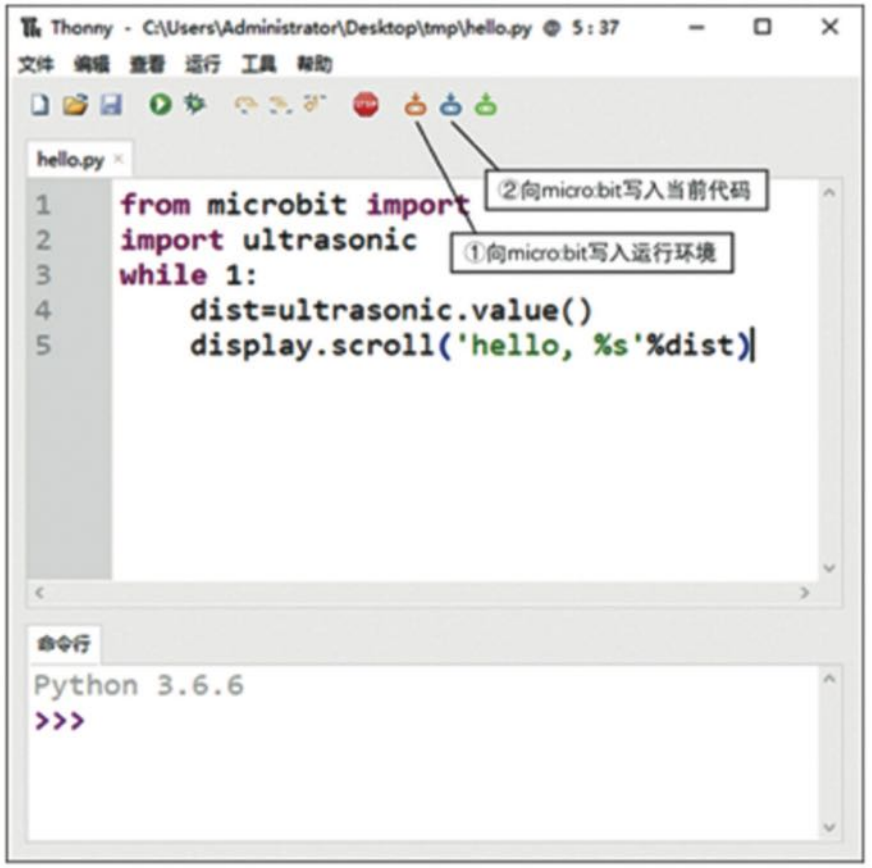
使用方式：

1）窗口分为两栏，左侧可以输入需要转换的原始文本，右侧可以输入python代码可以接收的字节串对象格式

2）“文本解码为GB2312”按钮可以将左侧文本框内内容逐行转换为字节串并在右侧输出，可以用于OLED模块的显示

3）“GB2312编码为文本”按钮可以将右侧每行的字节串对象转换为文字；若某行转换失败则会留空

6.3 输入和运行程序



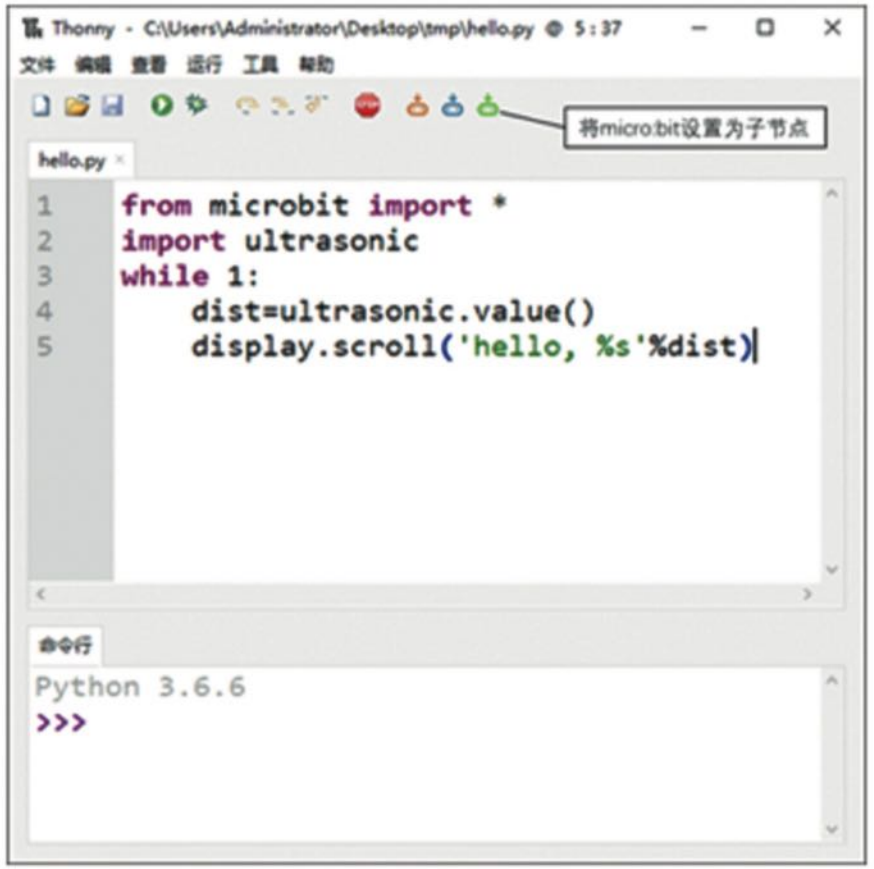
1)连接micro:bit，并点击橙色按钮写入运行环境；若该micro:bit已经写入过运行环境则此步不必要

2)编写代码并保存

3)点击蓝色按钮，将当前窗口内代码写入micro:bit中

4)重新启动micro:bit，运行代码

6.4 无线联网子节点配置



1)编写主节点控制代码并写入将作为主节点的micro:bit

2)连接将作为子节点的micro:bit

3)点击绿色按钮，将子节点代码写入该micro:bit

4)将子节点连接至主板，设置主节点与子节点的拨码开关，并启动主节点与子节点

6.6 dxkStick套件模块介绍

6.6.1 【DSK000】扩展板mb模块

模块调用方法：import mb

序号	属性/接口名称	功能和返回值说明
1	remote_on(short=1)	启动远程模式，short 表示是否为短路模式
2	get_type(addr)	获取指定插槽或 ID 模块的类型；返回 int 范围为 1-8
3	get_state(addr)	获取指定插槽或 ID 模块的状态；返回 int，应为 1
4	get_id(addr)	获取指定插槽模块的 ID；返回长度为 8 的 str

6.6.2 【DSK001】显示屏oled模块

模块调用方法：import oled

序号	属性/接口名称	功能和返回值说明
1	show(y, x, string, addr=None)	在显示屏指定行、列显示指定的文字内容
2	clear(addr=None)	清空显示屏内容

6.6.3 【DSK002】温湿度temp_humi模块

模块调用方法：import temp_humi

序号	属性/接口名称	功能和返回值说明
1	temp(addr=None)	获取传感器温度读数；返回 int 单位为℃
2	humi(addr=None)	获取传感器湿度读数；返回 int 单位为百分比(%)
3	temp_humi(addr=None)	获取传感器温度、湿度读数；返回长度为 2 的 tuple，分别为温度与湿度，含义与上文相同

6.6.4 【DSK003】LED灯led模块

模块调用方法：import led

序号	属性/接口名称	功能和返回值说明
1	on(addr=None)	打开灯泡
2	off(addr=None)	关闭灯泡

6.6.5 【DSK004】电位器poten模块

模块调用方法：import poten

序号	属性/接口名称	功能和返回值说明
1	value(addr=None)	返回 int 范围为[0, 4095]，分别对应旋钮两侧旋转极限

6.6.6 【DSK005】光敏电阻light模块

模块调用方法：import light

序号	属性/接口名称	功能和返回值说明
1	value(addr=None)	获取电阻读数；返回 int 范围为[0, 4095]，其中 0 对应最亮，4095 对应最暗

6.6.7 【DSK006】声音mic模块

模块调用方法：import mic

序号	属性/接口名称	功能和返回值说明
1	value(addr=None)	获取声音强度；返回 int 范围为[0, 4095]

6.6.8 【DSK007】手柄joypad模块

模块调用方法：import joypad

序号	属性/接口名称	功能和返回值说明
1	values(addr=None)	返回长度为 2 的元组，分别为 keys 与 stickxy 函数返回值，详见后文
2	keys(addr=None)	返回长度为 5 的元组，其值为 0 或 1，分别对应摇杆按键及右侧上、下、左、右按键是否按下
3	stickxy(addr=None)	返回长度为 2 的元组(x, y)，其中两个 int 分别对应摇杆横向、纵向相对自然状态偏移量（绝对值不大于 2048）；摇杆向右时 x 增加，向下时 y 增加

6.6.9 【DSK008】超声波测距ultrasonic模块

模块调用方法：import ultrasonic

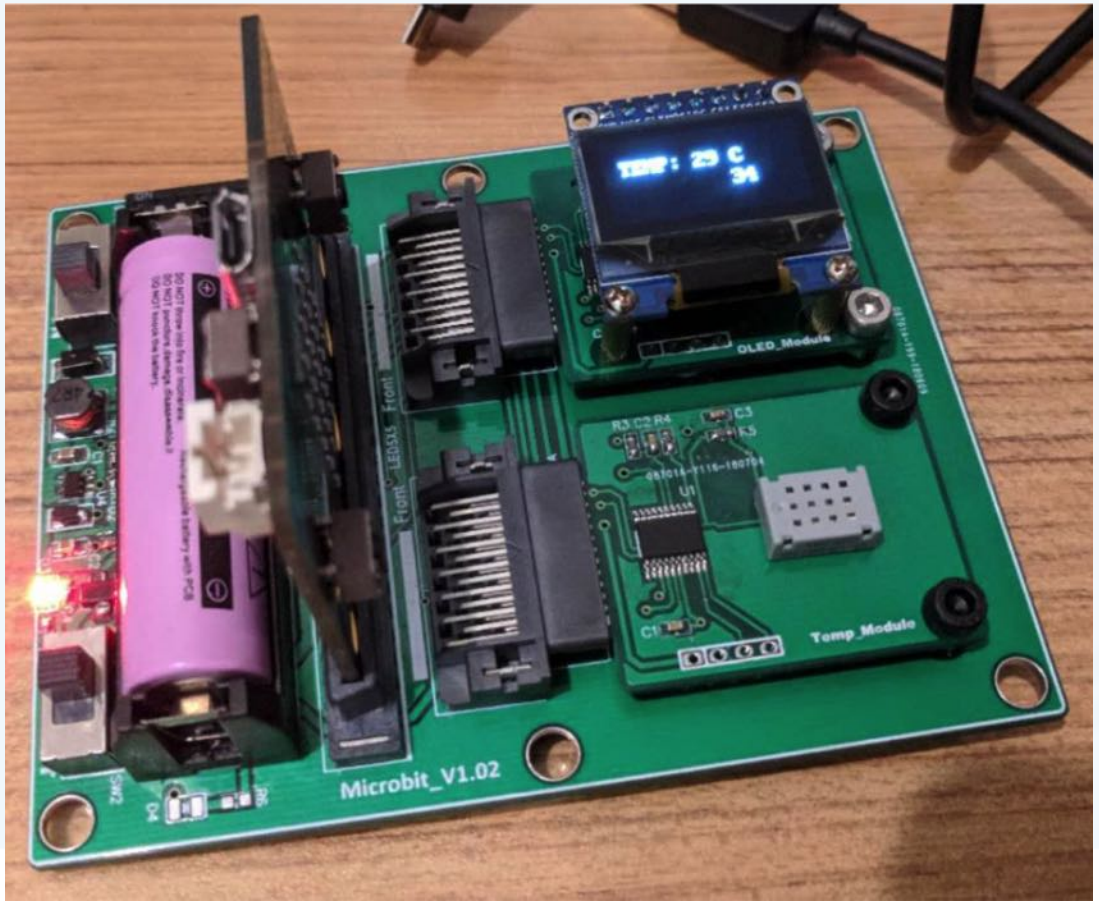
序号	属性/接口名称	功能和返回值说明
1	value(addr=None)	获取距离读数；返回 int 范围为[0, 400]，单位为 cm

6.6.10 无线通信mb_node模块

模块调用方法：import mb_node

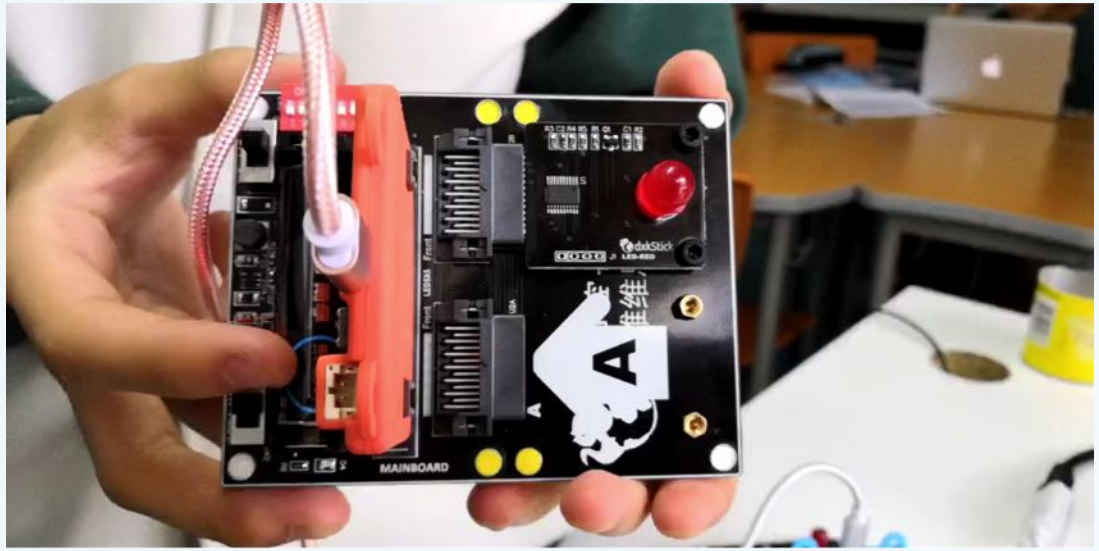
序号	属性/接口名称	功能和返回值说明
1	show(img, grp=-1)	使频道内指定组子节点显示指定图片
2	scroll(text, wait=0, grp=-1)	使频道内指定组子节点滚动显示指定文字
3	button(btn, grp=-1)	获取频道内指定组子节点对应按钮按下次数；返回元组包含所有响应节点的结果，其中元素为长度为 2 的元组：（按钮按下次数，该节点组别）
4	play(mid, wait=0, grp=-1)	使频道内指定组子节点播放指定音乐

实例：读取温度并显示（microbit编程）




```
from microbit import *
port_A = 0x16
port_B = 0x17

def get_temp(slot):
    i2c.write(slot, b'get_temp')
    b= i2c.read(slot, 1)
    v= int.from_bytes(b, 'big')
    return v

def show(slot, r, c, s):
    v= b'DisplayGB2312,%d,%d,%s    ' % (r, c, s)
    i2c.write(slot, v)
    sleep(50)
    return

c = 0
while True:
    t = get_temp(port_A)
    show(port_B, 2, 10, 'TEMP: %s C' % t)
    show(port_B, 4, 80, c)
    sleep(2000)
    c+= 1
```

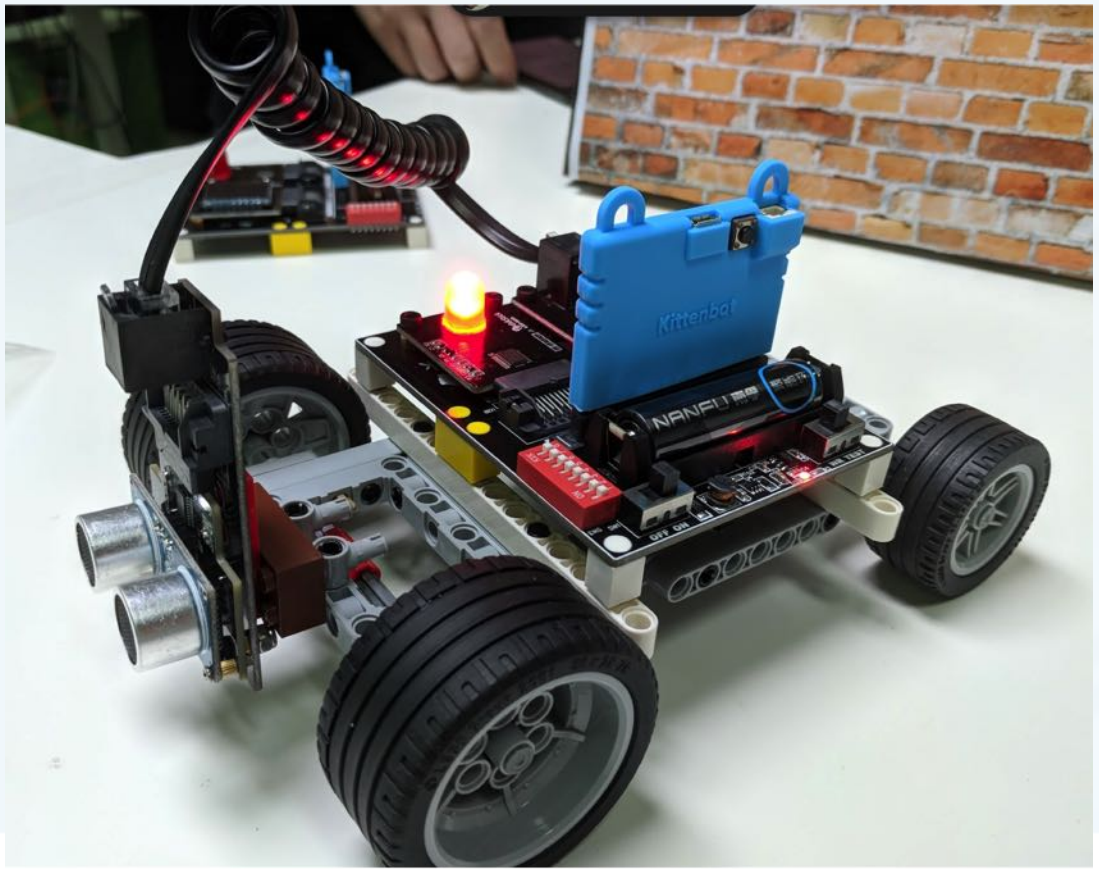
教学实例：SOS莫尔斯码（扩展库编程）




```
29  while True:
30      s()  # 字符s的莫尔斯码
31      sleep(150)  # 字符与字符之间等待150毫秒
32      o()  # 字符o的莫尔斯码
33      sleep(150)  # 字符与字符之间等待150毫秒
34      s()  # 字符s的莫尔斯码
35      sleep(150)  # 字符与字符之间等待150毫秒
36      sleep(350)  # 单词之间等待350毫秒
```

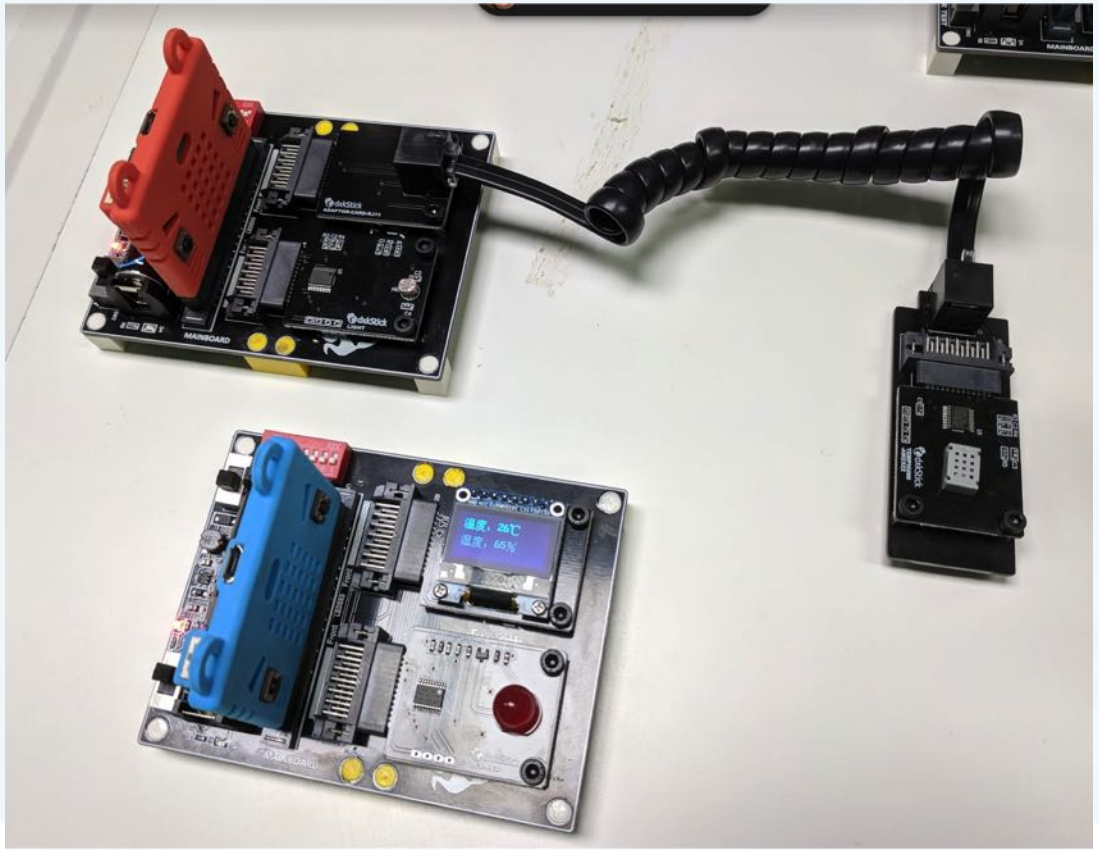
```
13  def s():  # 字符s的莫尔斯码表示
14      for i in range(3):  # 滴循环三次
15          music.pitch(1000, 50, wait=0)  # 蜂鸣 频率 持续时间 等待时间
16          led.on()  # 点亮led
17          sleep(50)  # 延迟50毫秒
18          led.off()  # 关闭led
```

实例：倒车雷达




```python
# 倒车雷达

# 简介:
# 模拟倒车雷达功能，通过超声测距模块测量与障碍物的距离
# 并根据距离远近发出不同频率的蜂鸣声与LED灯闪烁提示。

# 硬件模块:
# micro:bit×1；主板×1；延长插槽×1
# 模块×2：超声测距、LED灯泡

from microbit import *
import music
import ultrasonic, led  # 模块控制库

# 初始化LED灯闪烁系统
light_on = 0
ltimer = 0
led.off()

# 初始化测距记录系统
dist = ultrasonic.value()
update_timer = 0
```

实例：无线传输温度值



```python
# 远程测量温湿度、亮度

# 简介:
# 主节点读取子节点上传感器的读数并将结果显示在自身的元件上:
# 温湿度传感器的读数将直接显示于OLED显示屏内，光敏电阻则会在光强低于一定阈值

# 硬件模块:
# micro:bit×2; 主板×2
# 模块×4: （主节点）OLED显示屏、LED灯泡; （子节点）温湿度传感器、光敏电阻

import mb,oled,light,led,temp_humi # 主板、模块控制库
from microbit import *
import music

# 初始化
mb.remote_on()# 启动远程模式
led_on=0
oled.clear()
led.off()
timer=0
light_read=0
tmp,hum='--','--'
```

dxkStick开源仓库（开发环境/案例）

- https://github.com/chbpku/dxkStickIDE