好搭 bit 的 python 编程指令介绍及示例

1. 开头代码：

导入 microbit python 库：from microbit import *

2. 显示：

LED 显示屏通过 display 对象显示。

指令：

点阵屏显示：display.show()

示例：

点阵屏显示数字

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

点阵屏显示字符串

```
1. from microbit import *
2. display.show('a')
```

指令：

点阵屏清除：display.clear()

示例：

点阵屏数字 1 闪烁,间隔 1 秒。

```
from microbit import *
while True:
    display.show(1)
    sleep(1000)
    display.clear()
    sleep(1000)
```

指令：

点阵屏滚动显示：display.scroll()

示例：

点阵屏滚动显示 hello world!

```
from microbit import *
display.scroll("hello world!")
```

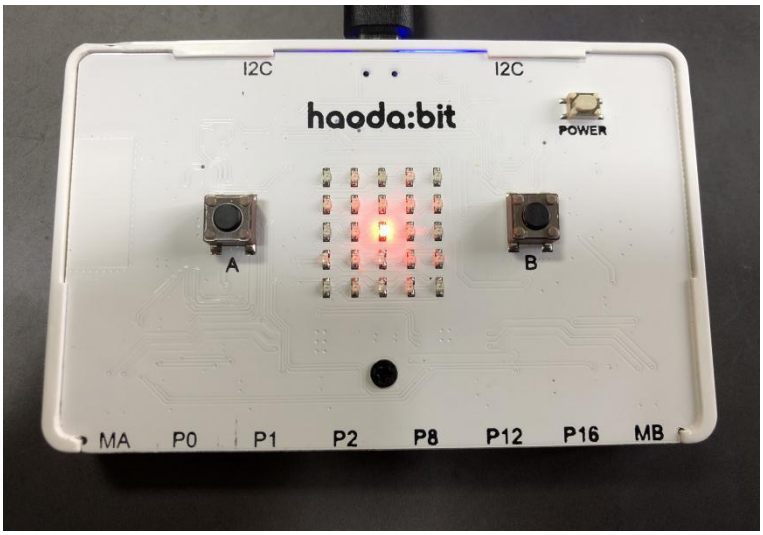
指令：

点阵屏坐标为（x，y）的点亮起：display.get_pixel(x, y)
点阵屏坐标为（x，y）的点亮起并设置亮度：display.set_pixel(x, y,val)

示例：

点阵屏坐标（2,2）点以亮度 8 亮起。

```
1. from microbit import *
2. display.set_pixel(2,2,8)
```



3. 板载的两个按键 A、B

button_a
Button_b

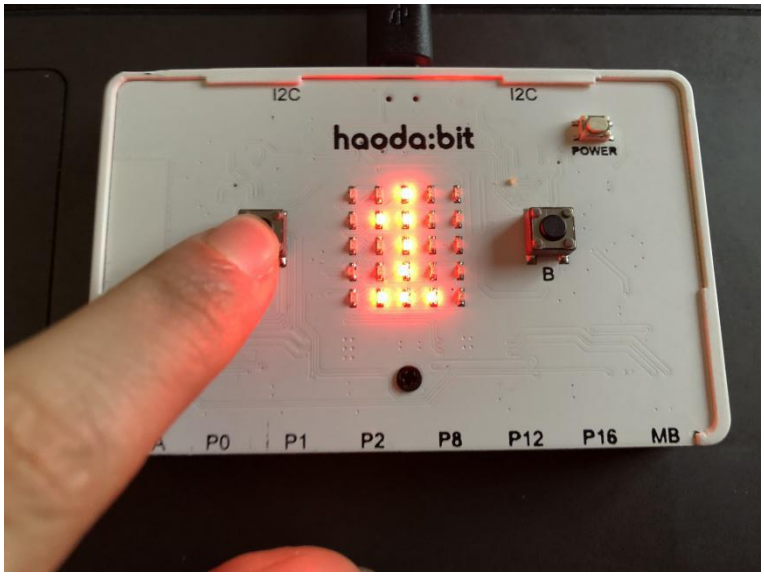
指令：

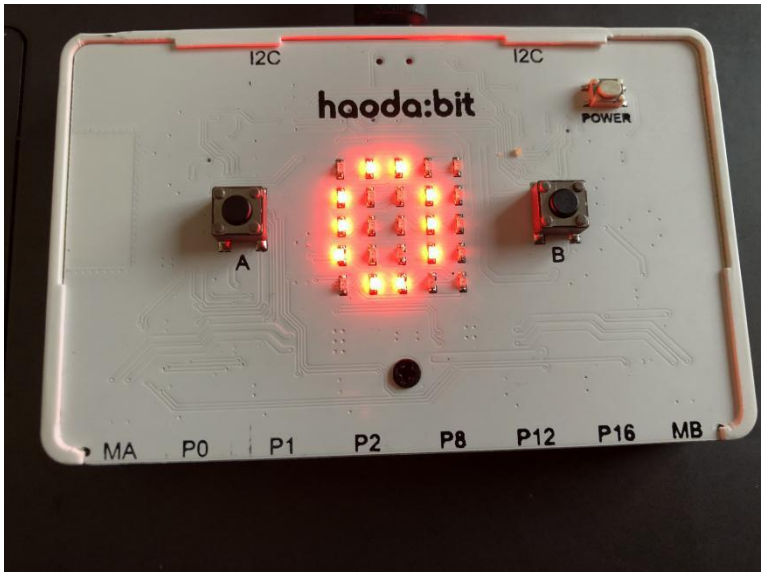
当按键 A 被按下：button_a.is_pressed()
当按键 B 被按下：button_b.is_pressed()

示例：

如果按键 A 被按下，显示 1，否则，显示 0。

```
from microbit import *
while True:
    if button_a.is_pressed():
        display.show(1)
    else:
        display.show(0)

display.clear()
```





4. 引脚

好搭 bit 外接引脚有 P0、P1、P2、P8、P12、P16。6 个引脚都有数字和模拟的输入输出功能。

指令：

数字输出（0/1）：pin.write_digital(value)
如果 value 是 1 设置引脚为高，如果是 0 设置为低。

数字读取：pin.read_digital()
如果引脚高了返回 1，低了返回 0。

模拟输出（0~1023）：pin.write_analog(value)
在引脚上输出一个 PWM 信号，其占空比与所提供的 value 成比例。 value 可以是一个整数，也可以是一个 介于 0（0％占空比）和 1023（100％占空比）之间的浮点数。

模拟读取：pin.read_analog()
读取施加于引脚的电压，并将其作为整数返回 0（意为 0V）和 1023（意为 3.3V）之间。

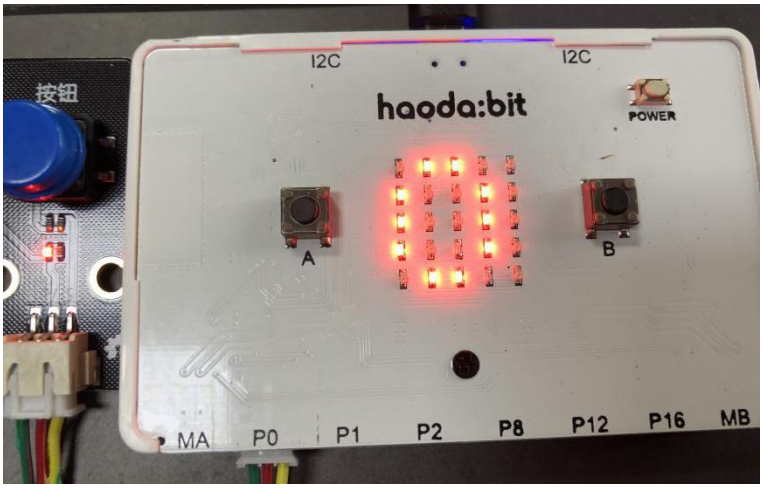
PWM 输出（ms）：pin.set_analog_period(value)
将输出的 PWM 信号的周期设置为 value 毫秒

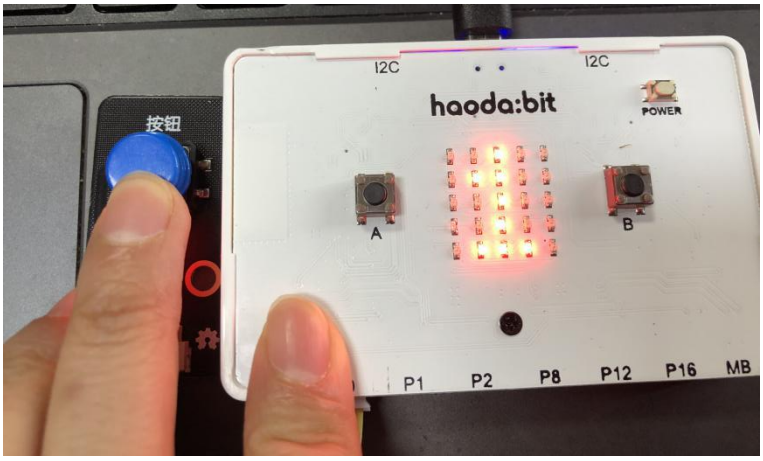
PWM 输出（us）：pin.set_analog_period_microseconds(value)
将输出的 PWM 信号的周期设置为 value 微秒

示例：

在 P0 连接单按键模块，点阵屏显示读取到的数字量。

```
1. from microbit import *
2. while True:
3.     display.show(pin0.read_digital())
4.     sleep(1000)
```





在 P0 连接 LED 模块，LED 灯闪烁。

```
from microbit import *
while True:
    pin0.write_digital(1)
    sleep(1000)
    pin0.write_digital(0)
    sleep(1000)
```

5. 图像：

通过 image 对象。

指令：

直接调用 Image 里的图案（举例四个）：
Image.HEART
Image.HEART_SMALL
Image.HAPPY
Image.SAD

示例：

点阵屏显示跳动的爱心

```
from microbit import *

while True:
    display.show(Image.HEART)
    sleep(1000)
    display.show(Image.HEART_SMALL)
    sleep(1000)
```

指令：

创建一个图案，分别为 1~5 行，数据为灯的亮度，0 即不亮。
image = Image('90009:09090:00900:09090:90009:')

示例：

点阵屏显示沙漏图案。

```
from microbit import *

Image1=Image("99999:09990:00900:09990:99999")
display.show(Image1)
```

6. 加速度计：

通过 accelerometer 对象访问。

指令：

获取 X 方向加速度：accelerometer.get_x()
获取 Y 方向加速度：accelerometer.get_y()
获取 Z 方向加速度：accelerometer.get_z()
获取三个方向整合的加速度：accelerometer.get_values()

示例：

点阵屏每隔一秒刷新读取到的 X 方向上的加速度值。

```
from microbit import *

while True:
    display.show(accelerometer.get_x())
    sleep(1000)
```

指令：

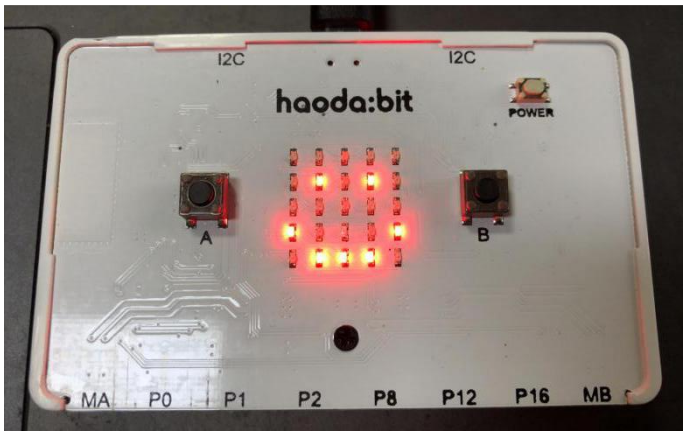
获取当前的手势名字：accelerometer.current_gesture()
判断当前手势是否为 name：accelerometer.is_gesture(name)
判断之前手势是否为 name：accelerometer.was_gesture(name)
获取所有做过的手势：accelerometer.get_gesture()

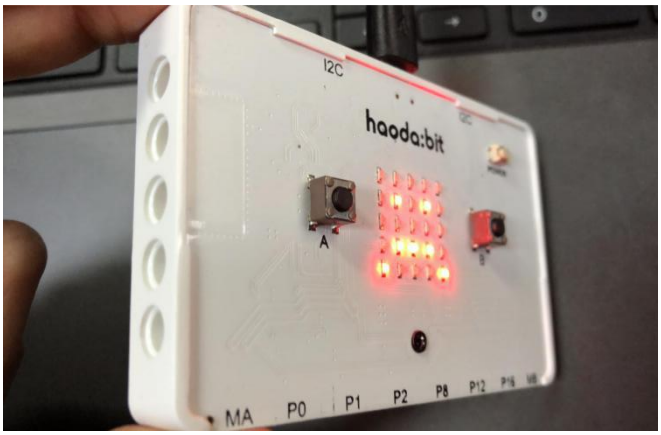
可识别手势为：向上（up）、向下（down）、向左（left）、向右（right）、正面朝上（face up）、正面朝下（face down）、自由落体（free fall）、3g（threeg）、6g（sixg）、8g（eightg）、摇动（shake）

示例：

当手势为正面朝上时，点阵屏显示笑脸；否则，显示哭脸。

```
from microbit import *

while True:
  gesture = accelerometer.current_gesture()
  if gesture == "face up":
    display.show(Image.HAPPY)
  else:
    display.show(Image.SAD)
```





摇晃好搭 bit，显示笑脸；否则，显示哭脸。

```
from microbit import *

while True:
    if accelerometer.is_gesture("shake"):
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)

```

7.电子罗盘：

电子罗盘通过 compass 对象访问。

指令：

校准指南针：compass.calibrate()
指南针已被校准：compass.is_calibrated()
重置电子罗盘：compass.clear_calibration()
获取指南针朝向：compass.heading()
获取周围磁场强度：compass.get_field_strength()

示例：

指南针程序，点阵屏一道光束指向北方。

```
from microbit import *

compass.calibrate()

while True:
    needle = ((15 - compass.heading()) // 30) % 12
    display.show(Image.ALL_CLOCKS[needle])
```

8. 随机数：

通过 random 对象：import random

指令：

随机返回一个 a 和 b 之间的数：random.randint(a,b)

示例：

点阵屏每隔一秒随机显示一个 1-10 之间的数。

```
from microbit import *
import random

while True:
    display.show(random.randint(1,10))
    sleep(1000)
```

9. 音乐

通过 music 对象：import music

指令：

播放音乐：music.play(music)
暂停音乐：music.stop()
可直接播放编码，也可直接调用 music 里的音乐（举例五个）
-DADADADUM - 贝多芬-C 小调第五交响曲开场。
-ENTERTAINER - Scott Joplin 的 Ragtime 经典“The Entertainer”的开场片段。
-PRELUDE - 巴赫的 48 首 C 大调前奏曲和赋格曲的前奏曲开场。
-ODE - 贝多芬 D 小调第九交响曲《欢乐颂》主题
-NYAN- Nyan Cat 主题

示例：

将蜂鸣器模块连接到 P0 引脚，播放 Nyan Cat 主题曲

```
from microbit import *
import music

```

```
4. music.play(music.NYAN)
```

将蜂鸣器模块连接到 P0 引脚，播放 note 编码曲目

```
1. from microbit import *
2. import music
3. 
4. notes = [
5.     'c4:1', 'e', 'g', 'c5', 'e5', 'g4', 'c5', 'e5', 'c4', 'e', 'g', 'c5', 'e5', 'g4', 'c5', 'e5',
6.     'c4', 'd', 'g', 'd5', 'f5', 'g4', 'd5', 'f5', 'c4', 'd', 'g', 'd5', 'f5', 'g4', 'd5', 'f5',
7.     'b3', 'd4', 'g', 'd5', 'f5', 'g4', 'd5', 'f5', 'b3', 'd4', 'g', 'd5', 'f5', 'g4', 'd5', 'f5',
8.     'c4', 'e', 'g', 'c5', 'e5', 'g4', 'c5', 'e5', 'c4', 'e', 'g', 'c5', 'e5', 'g4', 'c5', 'e5',
9.     'c4', 'e', 'a', 'e5', 'a5', 'a4', 'e5', 'a5', 'c4', 'e', 'a', 'e5', 'a5', 'a4', 'e5', 'a5',
10.     'c4', 'd', 'f#', 'a', 'd5', 'f#4', 'a', 'd5', 'c4', 'd', 'f#', 'a', 'd5', 'f#4', 'a', 'd5',
11.     'b3', 'd4', 'g', 'd5', 'g5', 'g4', 'd5', 'g5', 'b3', 'd4', 'g', 'd5', 'g5', 'g4', 'd5', 'g5',
12.     'b3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5', 'b3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5',
13.     'b3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5', 'b3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5',
14.     'a3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5', 'a3', 'c4', 'e', 'g', 'c5', 'e4', 'g', 'c5',
15.     'd3', 'a', 'd4', 'f#', 'c5', 'd4', 'f#', 'c5', 'd3', 'a', 'd4', 'f#', 'c5', 'd4', 'f#', 'c5',
16.     'g3', 'b', 'd4', 'g', 'b', 'd', 'g', 'b', 'g3', 'b3', 'd4', 'g', 'b', 'd', 'g', 'b'
17. ]
18. 
19. music.play(notes)
```

指令：

更改音乐节奏：music.set_tempo(ticks=4, bpm=120)
music.set_tempo() - 重置节奏为默认值 ticks = 4, bpm = 120
music.set_tempo(ticks=8) - 改变每个节拍的“定义”
music.set_tempo(bpm=180) - 只改变节奏

以指定毫秒数的整数频率播放音调：music.pitch(frequency，duration=-1)
重置音乐属性：music.reset()

示例：

将蜂鸣器连接到 P0，蜂鸣器响起。

```
from microbit import *
import music

while True:
    for freq in range(880, 1760, 16):
        music.pitch(freq, 6)
```

10.蓝牙

虽然好搭 bit 具备作为低功耗蓝牙设备工作的硬件，但是它只有 16kRAM，蓝牙堆栈就要占用 12kRAM，因此没有足够的空间运行 MicroPython。

11.无线

通过 radio 对象：import radio

常量：
每秒 256Kbit 的吞吐量：radio.RATE_250KBIT
每秒 1Mbit 的吞吐量：radio.RATE_1MBIT
每秒 2Mbit 的吞吐量：radio.RATE_2MBIT

指令：

打开无线：radio.on()
关闭无线：radio.off()
发送含有字节的信息：radio.send_bytes(message)
发送字符串消息：radio.send(message)
接收收到的信息队列中的下一个消息（返回所有发送）：radio.receive()
接收收到的信息队列中的下一个消息：radio.receive_bytes()
接收收到的信息队列中的下一个消息到数组 buffer：radio.receive_bytes_into(buffer)

示例：

上面引用的事件会发生在事件循环中。首先，它会检查按钮 A 是否被按下，如果是，则使用无线电发送信息“闪（flash）”。然后它使用 radio.receive() 从信息队列中读取信息。如果有信息，它会随机进行短时间的休息（以使显示更有趣），并使用 display.show() 为萤火虫发光设置动画。最后，它会选择一个随机数，这样它就有 1/10 的机会将“闪”信息重新传递给其他任意设备。如果它决定重播，那么在再次发送“闪”信号之前，会等待半秒钟（所以最初的闪光信息有机会消失）。因为这段代码被包含在 while True 代码块中，所以它会循环回到事件循环的开始，并且永久地重复这个过程。

```
import radio
import random
from microbit import display, Image, button_a, sleep

flash = [Image().invert()*(i/9) for i in range(9, -1, -1)]

radio.on()

while True:

    if button_a.was_pressed():
        radio.send('flash')
    incoming = radio.receive()

    if incoming == 'flash':

        sleep(random.randint(50, 350))
        display.show(flash, delay=100, wait=False)

        if random.randint(0, 9) == 0:
            sleep(500)
            radio.send('flash')
```

现象：使用 9 个好搭 bit，好搭 bit 点阵屏随机亮起。

11.UART

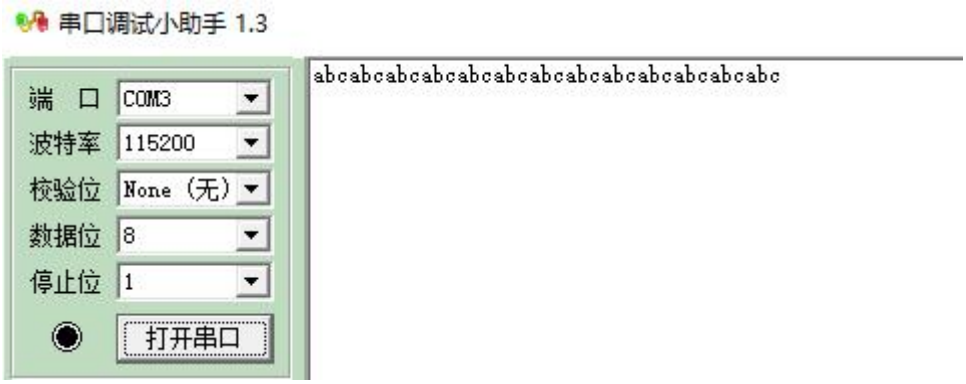
指令：

初始化串口通信：
microbit.uart.init(baudrate=9600, bits=8, parity=None, stop=1, *, tx=None, rx=None)
baudrate 定义通信速度（设置波特率），bits 定义了正在传输的字节大小，tx 和 rx 可指定，若没有指定则为 USB 串口。
将字节的缓冲写入总线：uart.write(buf)
读取一行的数据：uart.readline()
将字节读入 buf：uart.readinto(buf[,nbytes])
尽可能多的读取数据：uart.readall()
读取字符：uart.read([nbytes])
判断是否读到字符：uart.any()

示例：

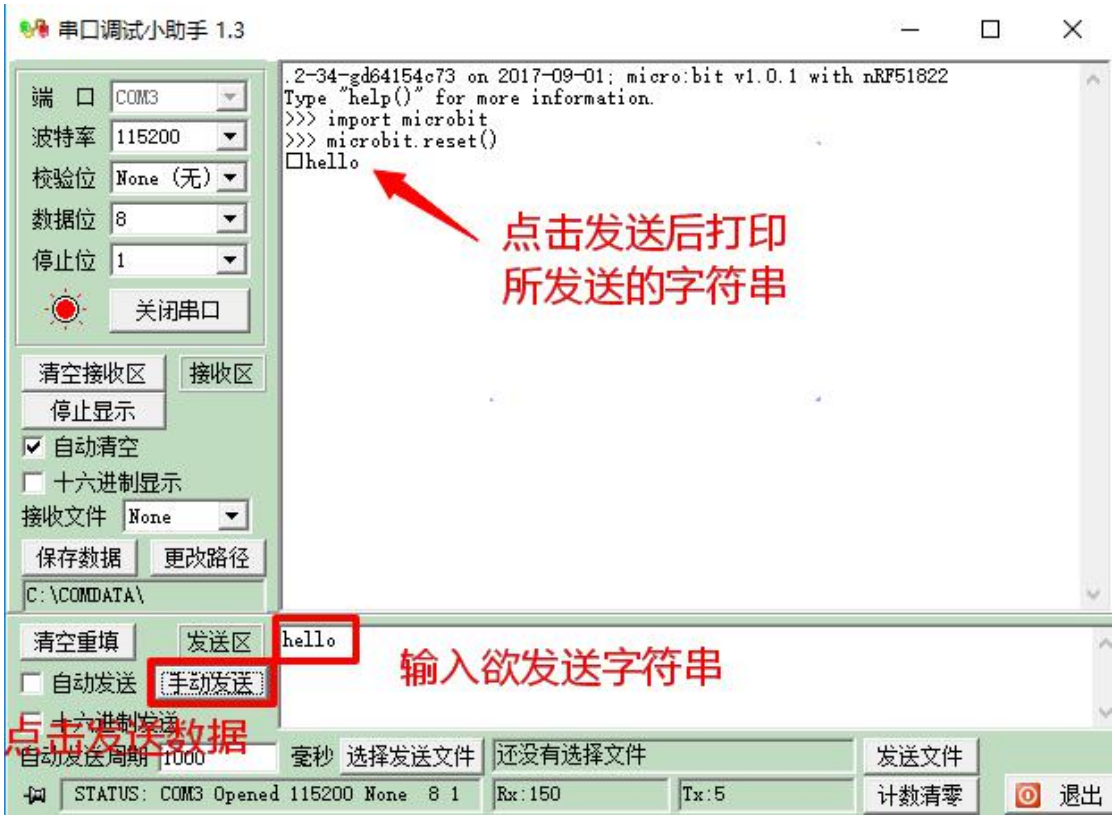
1. 打开串口助手，设置波特率为 115200，打开串口，接收区每隔一秒接收到“abc”。

```
from microbit import *

uart.init(115200)
while True:
    uart.write('abc')
    sleep(1000)
```



2. 打开串口助手，设置波特率为 115200，打开串口，在发动送区输入字符“hello”，点击手动发送，接收区打印出“hello”。

```
from microbit import *

uart.init(115200)
while True:
    if uart.any():
        tt = uart.read()
        uart.write(tt)
```



12.I2C

指令：

初始化 I2C：microbit.i2c.init()
使用 7 位寻址 addr 从设备中读取 n 个字节：microbit.i2c.read(addr,n,repeat=False)
若写 repeat=true，则不发送停止位
使用 7 位寻址 addr 将 buf 中的字节写入设备：microbit.i2c.write(addr,buf,repeat=False)
若写 repeat=true，则不发送停止位

示例：

将 LCD1602 显示模块连接到 I2C 接口，下载程序，屏幕显示“Hello haohaodada!”

```
from microbit import *

LCD_I2C_ADDR=59

class LCD1620():
    def __init__(self):
        self.buf = bytearray(1)
        self.BK = 0x08
        self.RS = 0x00
        self.E = 0x04
        self.setcmd(0x33)
        sleep(5)
        self.send(0x30)
        sleep(5)
        self.send(0x20)
        sleep(5)
        self.setcmd(0x28)
        self.setcmd(0x0C)
        self.setcmd(0x06)
        self.setcmd(0x01)
        self.version='1.0'

    def setReg(self, dat):
        self.buf[0] = dat
        i2c.write(LCD_I2C_ADDR, self.buf)
        sleep(1)

    def send(self, dat):
```

```
        d=dat&0xF0
        d|=self.BK
        d|=self.RS
        self.setReg(d)
        self.setReg(d|0x04)
        self.setReg(d)

    def setcmd(self, cmd):
        self.RS=0
        self.send(cmd)
        self.send(cmd<<4)

    def setdat(self, dat):
        self.RS=1
        self.send(dat)
        self.send(dat<<4)

    def clear(self):
        self.setcmd(1)

    def backlight(self, on):
        if on:
            self.BK=0x08
        else:
            self.BK=0
        self.setdat(0)

    def on(self):
        self.setcmd(0x0C)

    def off(self):
        self.setcmd(0x08)

    def char(self, ch, x=-1, y=0):
        if x>=0:
            a=0x80
            if y>0:
                a=0xC0
            a+=x
            self.setcmd(a)
        self.setdat(ch)

    def puts(self, s, x=0, y=0):
        if len(s)>0:
```

```
            self.char(ord(s[0]),x,y)
            for i in range(1, len(s)):
                self.char(ord(s[i]))

tt = LCD1620()
tt.puts("Hello haohaodada!")
```