核心單元
類庫參考

發佈時間 2019/04/23

修訂歷史

修訂日期	修訂內容
2019/4/23	首次發佈
2019/9/20	添加主機板模組資訊
2020/2/19	Mu Editor 現在是推薦的程式設計環境 將模組資訊添加到類表附錄

目錄

1. 入門 1
2. Studuino：bit 1
   2.1. Studuino：bit 佈局 1
   2.2. MicroPython 2
3. 開發環境 2
4. Studuino：bit 類庫 3
   4.1. StuduinoBitDisplay 類 3
      4.1.1. 構造函數 3
      4.1.2. 指示燈設置 4
      4.1.3. 從 LED 中檢索數據 4
      4.1.4. 動畫 5
      4.1.5. 顯示電源設置 6
   4.2. StuduinoBitImage 類 7
      4.2.1. 構造函數 8
      4.2.2. 查找圖像大小 9
      4.2.3. 圖像設置 9
      4.2.4. 查找圖像數據 10
      4.2.5. 操作圖像 11
      4.2.6. 包含的圖像 12
   4.3. StuduinoBitBuzzer 類 13
      4.3.1. 構造函數 13
      4.3.2. 調整聲音 13
      4.3.3. 發佈 PWM 13
   4.4. StuduinoBitButton 類 14
      4.4.1. 構造函數 14
      4.4.2. 查找按鈕狀態 14
   4.5. StuduinoBitLightSensor 類 15
      4.5.1. 構造函數 15
      4.5.2. 查找亮度 15
   4.6. StuduinoBitTemperature 類 16

4.6.1. 構造函數 16
4.6.2. 查找溫度 16
4.7. StuduinoBitAccelerometer 類 17
4.7.1. 構造函數 17
4.7.2. 加速度計設置 17
4.7.3. 尋找加速度 17
4.8. StuduinoBitGyro 類 18
4.8.1. 構造函數 18
4.8.2. 陀螺儀設置 18
4.8.3. 查找角速度 18
4.9. StuduinoBitCompass 類 19
4.9.1. 構造 函數 19
4.9.2. 校準 19
4.9.3. 查找路線 19
4.9.4. 查找指南針值 19
4.10. StuduinoBitTerminal 類 20
4.10.1. 構造函數 20
4.10.2. 數位輸入/輸出 20
4.10.3. 數位輸入/輸出和模擬輸入 21
4.10.4. 數位和模擬輸入 21
5. 附錄 22
5.1. 板模組 22
5.2. 聲音輸出 23
5.3. 包含的圖像 24
5.4. 類表 26

1. 入門

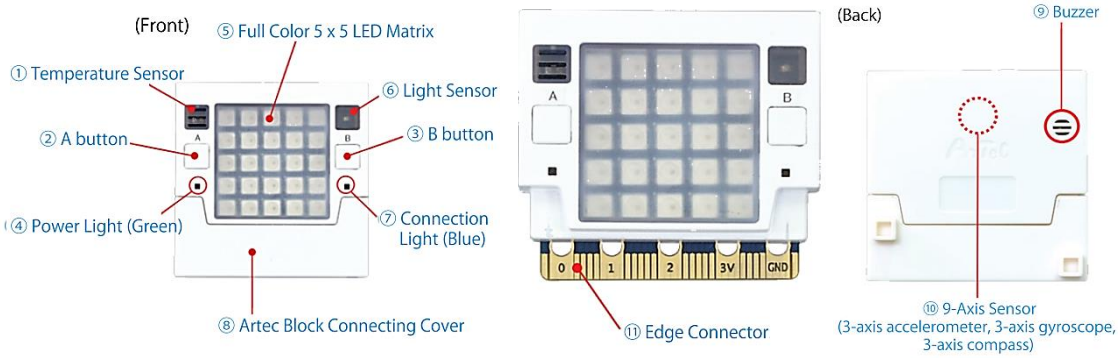
本手冊是 ArtecRobo2.0 Studuino 類庫的參考指南。使用本手冊需要事先瞭解 Python 的基礎知識。

2. Studuino：bit

Studuino：bit 是一種配備微控制器，感測器，LED 和其他部件的計算設備。 MicroPython 是與 Studuino：bit 兼容的程式設計環境之一。

2.1. Studuino：bit 佈局

Studuino：bit 的佈局如下所示。



可與 Studuino：bit 類庫配合使用的硬體部件包括：① 溫度感測器、② A 按鈕、③ B 按鈕、⑤ 全彩色 5 x 5 LED 矩陣、⑥ 光感測器、⑨ 蜂鳴器、⑩ 9 軸感測器（3 軸加速度計、3 軸陀螺儀、3 軸羅盤）和 ⑪ 邊緣連接器。

從這裡開始，全彩色 5 x 5 LED 矩陣將被稱為 LED 顯示幕。9 軸感測器的三個部分（3 軸加速度計，3 軸陀螺儀，3 軸羅盤）將分別稱為加速度計，陀螺儀和地磁感測器。邊緣連接器將稱為埠。

2.2. MicroPython

MicroPython 是專門為微控制器開發的 Python 版本。程式設計語法與 Python 3.0 相同，但某些 Python 庫在 MicroPython 中不可用，因為它們不是為與微控制器的有限記憶體和 CPU 一起使用而設計的。但是，某些特定於 MicroPython 的庫（例如 GPIO 微控制器的庫）作為標準包含在內。

MicroPython 已被廣泛移植到不同的微控制器， Studuino：bit 庫使用 Studuino：bit 特定版本。

3. 開發環境

一些可以與 Studuino：bit 一起使用的 MicroPython 開發環境是 Mu Editor 和 uPyCraft。我們在以下 URL 的網站上提供了使用 Mu Editor 的安裝程式和說明：
https://www.artec-kk.co.jp/artecrobo2/en/software/python.php

4. Studuino：bit 類庫

Studuino：bit 類庫的結構如下。

機組	模組	類	硬體
pystubit	dsply	StuduinoBitDisplay	LED 顯示幕
	圖像	StuduinoBitImage	
	bzr	StuduinoBitBuzzer	蜂鳴器
	按鈕	StuduinoBitButton	A/B 按鈕
	感測器	StuduinoBitLightSensor	光感測器
		StuduinoBitTemperature	溫度感測器
		StuduinoBitAccelerometer	加速度計
		StuduinoBitGyro	陀螺儀
		StuduinoBitCompass	地磁感測器（指南針）
	終端	StuduinoBitTerminal	埠

導入板模組將允許您使用上述類中的物件（請參閱第 5.1

4.1. StuduinoBitDisplay 類

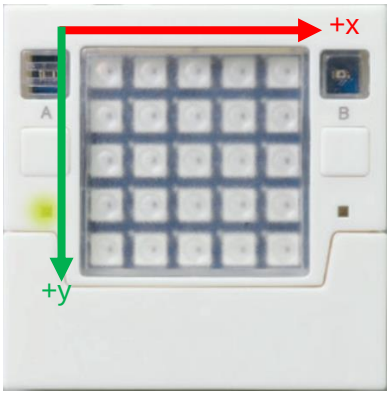
StuduinoBitDisplay 類用於操作 Studuino：bit 核心單元正面的 5 x 5 全彩 LED 顯示幕 。此類可用於使 LED 顯示圖像、動畫和文本。

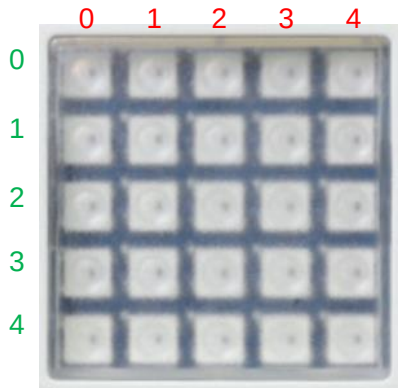
4.1.1. 構造函數

用它來製作一個操作 LED 顯示幕的物體。

```
from pystubit.dsply import StuduinoBitDisplay
display = StuduinoBitDisplay（）
```

4.1.2. 指示燈設置



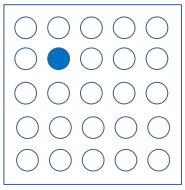


使用 set_pixel（x，y，顏色）的方法更改特定 LED 的設置。顯示器中的每個 LED 都由一組特定的 x-y 座標指定（見上圖）。 set_pixel 方法中的 x 參數選取 x 座標，y 選取 y 座標，顏色設置顏色。可以使用清單、元組或整數指定顏色參數。如果使用元組或清單，請使用 （R、G、B） 或 [R、G、B] 來設置紅色、綠色和藍色光照級別。如果使用整數，請使用十六進位代碼設置顏色，如下所示。您還可以使用 clear（） 方法將顯示器中所有 LED 的亮度設置為 0（關閉）。

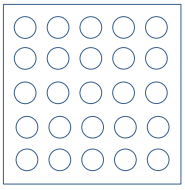
0 倍 ffffff
紅 綠 藍

```
from pystubit.dsply import StduinoBitDisplay
display = StduinoBitDisplay()
display.set_pixel(1, 1, (0, 0, 10))
display.clear()
```

運行這些方法中的每一種都應使 LED 顯示幕回應如下圖所示。



運行 set_pixel 方法



運行清除方法

4.1.3. 從 LED 中檢索數據

使用 get_pixel（x， y） 的方法從特定 LED 獲取資訊。將 x 和 y 參數設定為與顯示幕上 LED 的 x-y 座標相匹配，以元組格式找出 LED 的顏色。

```
from pystubit.dsply import StduinoBitDisplay
display = StduinoBitDisplay()
display.set_pixel(1, 1, (0, 0, 10))
c1 = display.get_pixel(0, 0)
c2 = display.get_pixel(1, 1)
```

變數 c1 從顯示幕上座標 （0， 0） 處的 LED 獲取數據 （0， 0， 0， 0），而變數 c2 將從座標 （1， 1） 獲取數據 （0， 0， 10）。

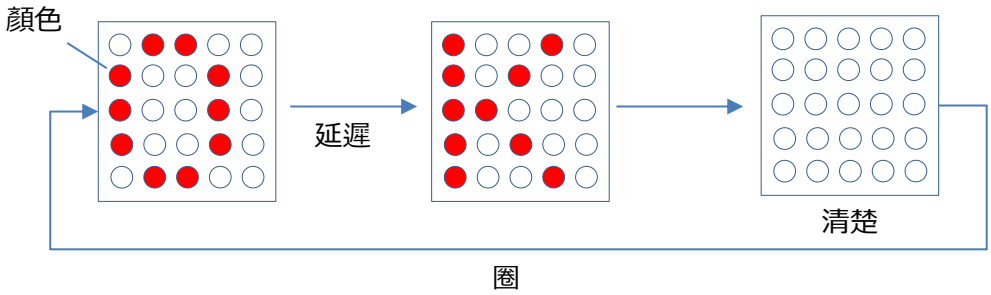
4.1.4. 動畫

方法 show（值、延遲、等待=真、循環=假、清除=假、顏色=無）可用 LED 顯示字母和圖像。

您在「值」參數中設定的任何數位或字母字串都將按 LED 順序顯示。如果在「值」參數中為圖像物件設置陣列（參見第 4.2 節 4.2

```
from pystubit.dsply import StduinoBitDisplay
display = StduinoBitDisplay()
display.show('OK', 100, wait=True, loop=True, clear=True, color=(10, 0, 0))
```

像這樣運行 show 方法會讓 LED 將字母 O 以紅色顯示 100 毫秒，然後字母 K 顯示 100 毫秒，最後清除顯示。此動畫將迴圈播放。

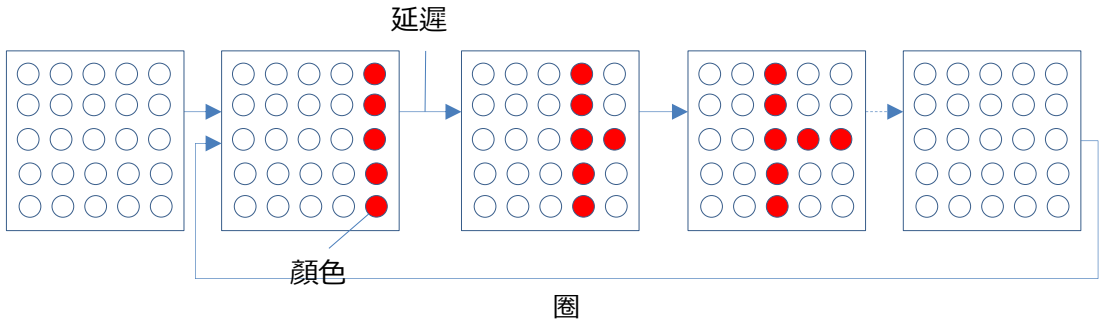


方法 display.scroll（字串、延遲、等待=真、循環=假、顏色=無）可用於使一系列字元在顯示器上滾動。

您可以使用「字串」參數設定要在滾動中顯示的字母和數位字串。使用 「延遲」 參數設置字串滾動的速度。如果「等待」參數設置為真，在顯示動畫完成之前，其他方法將被阻止。如果設置為假，其他方法則將在後台運行。如果「循環」參數設置為真，動畫則將循環播放。使用「顏色」參數可設置所顯示圖像的顏色。可以使用清單、元組或整數指定「顏色」參數。

```
from pystubit.dsply import StduinoBitDisplay
display = StduinoBitDisplay()
display.scroll('Hello', 100, wait=True, loop=True, color=(10, 0, 0))
```

運行這樣的滾動方法將使字串 Hello 以紅色在顯示幕上反覆滾動，如下所示。

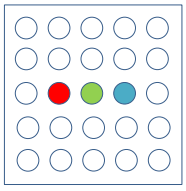


4.1.5. 顯示電源設置

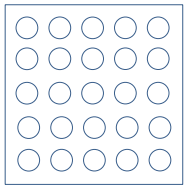
如果顯示器中所有 LED 的亮度都設置為 0，則將關閉顯示器的電源。也可以使用 off（） 方法將 LED 顯示器的電源關閉。同樣，on（） 方法也可用於打開 LED 顯示器的電源。您可以使用 is_on（） 方法檢查顯示器的電源狀態。

```
from pystubit.dsply import StuduinoBitDisplay
display = StuduinoBitDisplay()
display.set_pixel(1, 2, (10, 0, 0))
display.set_pixel(2, 2, (0, 10, 0))
display.set_pixel(3, 2, (0, 0, 10))
display.off()
ds1 = display.is_on()
display.on()
ds2 = display.is_on()
```

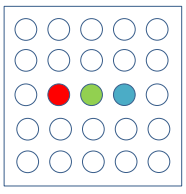
在下面的示例中，LED 顯示幕的座標 （1，2） 處的 LED 以紅色點亮，（2，2） 以綠色點亮，（3，2） 以藍色點亮。運行 off 方法將關閉 LED，運行 on 方法會再次打開它們。



運行 OFF 方法之前



運行 OFF 方法

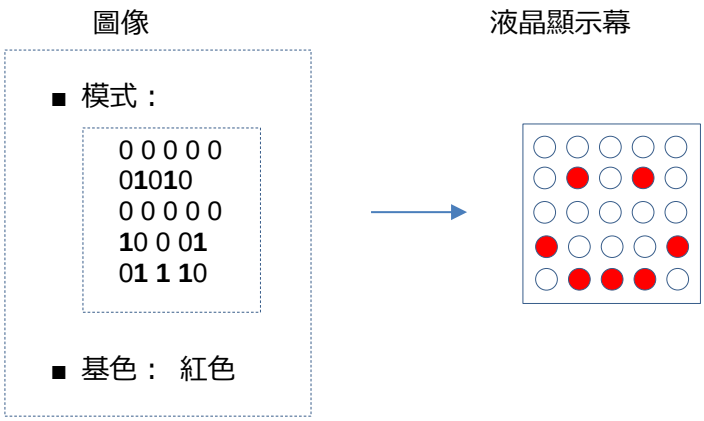


運行 ON 方法

變數 ds1 將設置為假，而 ds2 將設置為真。

4.2. StuduinoBitImage 類

使用 StuduinoBitImage 類可以輕鬆創建要使用 LED 顯示的圖像。圖像由圖案和顏色數據組成，供 LED 顯示。模式使用 0（表示 OFF）和 1（表示 ON）的系列編寫。請參閱下圖，瞭解圖像如何轉換為 LED 顯示幕的示例。



4.2.1. 構造函數

有四種方法可以製作圖像。

StuduinoBitImage（*，顏色）	使用所有 5 x 5 空間（全部 0）製作圖像。
StuduinoBitImage（字串，*，顏色）	從字串參數中的 0 和 1 字串創建一個模式。
StuduinoBitImage（寬度，高度，*，顏色）	使用指定寬度（寬度參數）和高度（高度參數）內的空格製作圖像。
StuduinoBitImage（寬度，高度，緩衝區，*，顏色）	使用指定圖案（緩衝區參數）在指定寬度（寬度參數）和高度（高度參數）內使用空格創建圖像。

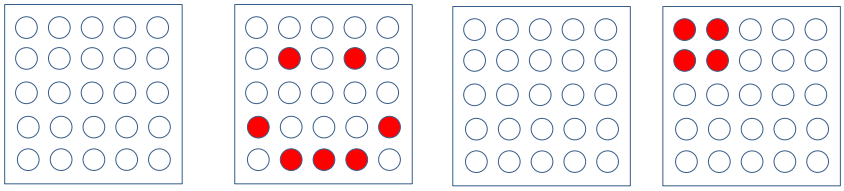
「顏色」參數可用於設置所有構造函數中的基色。可以使用清單、元組或整數指定「顏色」參數。如果尚未設置「顏色」變數，則將基色設為紅色 （31，0，0）。

```
from pystubit.image import StuduinoBitImage
img1 = StuduinoBitImage()
img2 = StuduinoBitImage('00000:'
                        '01010:'
                        '00000:'
                        '10001:'
                        '01110:')
img3 = StuduinoBitImage(2,2)
img4 = StuduinoBitImage(2,2, bytearray([1,1,1,1]))
```

您可以使用顯示物件顯示您創建的圖像，如下所示。

```
from pystubit.dsply import StuduinoBitDisplay
display.show(img1)
```

變數 img1、img2、img3 和 img4 將顯示在 LED 顯示幕上，如下所示。



使用 string 參數集的構造函數可以顯示在一行中，如下所示。

```
img2 = StuduinoBitImage('00000:01010:00000:10001:01110:')
```

4.2.2. 查找圖像大小

使用方法 width（） 和 height（） 尋找影像的大小。

```
from pystubit.image import StuduinoBitImage
img = StuduinoBitImage(2,3)
w = img.width()
h = img.height()
```

變數 w 將設置為 2，而 h 將設置為 3。

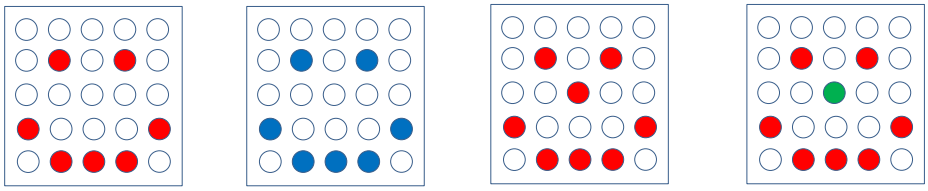
4.2.3. 圖像設置

使用 set_pixel（x， y， 值） 方法編輯圖像圖案。使用 x-y 座標指定圖元位置，就像顯示器中的 LED 一樣，然後將「值」參數設置為 0 或 1。還可以使用 set_pixel_color（x， y，顏色） 方法更改單個像素的顏色設置。指定圖元的座標（x 和 y 參數)， 在 顏色參數中設置顏色。

set_base_color（顏色）方法可用於設置基色。

```
from pystubit.image import StuduinoBitImage
original = StuduinoBitImage('00000:01010:00000:10001:01110:')
img1.set_base_color((0,0,10))
img2.set_pixel(2,2,1)
img3.set_pixel_color(2,2,(0,10,0))
```

原始變數 img1、img2 和 img3 將顯示在 LED 顯示幕上，如下所示。在 img1 中，您可以看到基色已用 set_base_color 方法更改為藍色（0，0，10)。在 img2 中,您可以看到（2，2） 處的圖元已用 set_pixel 方法設置為 1（使其顯示基色)。在 img3 中，您可以看到 （2，2） 處的圖元已用 set_pixel 方法更改為綠色 （0， 10， 0)。



4.2.4. 查找圖像數據

使用 get_pixel（x， y） 方法查找圖像的圖案資訊。指定圖元的座標（x 和 y 參數）將返回該圖元（0 或 1）的圖案數據。

使用 get_pixel_color（x， y） 方法查找圖像中單個像素的數據。指定圖元的座標（x 和 y 參數）以元組格式查找其顏色。

```
from pystubit.image import StuduinoBitImage
original = StuduinoBitImage('00000:01010:00000:10001:01110:')
v1 = img.get_pixel(0, 0)
c1 = img.get_pixel_color(0, 0)
v2 = img.get_pixel(1, 1)
c2 = img.get_pixel_color(1, 1)
```

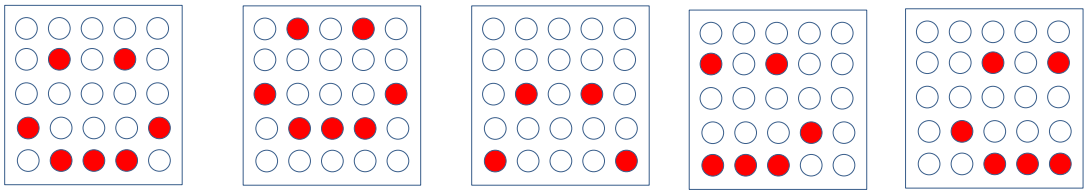
變數 v1 和 c1 從影像座標 （0， 0） 處的 LED 獲取圖案數據 （0） 和顏色數據 （0， 0），而變數 v2 和 c2 將從座標 （1， 1） 獲取圖案數據 （1） 和顏色數據 （31， 0， 0）。

4.2.5. 操作圖像

shift_up（n)、shift_down（n)、shift_left（n） 和 shift_right（n） 的方法可用於將圖像向上、向下、向左或向右移動 n 參數中指定的量。

```
from pystubit.dsply import StuduinoBitImage
original = StuduinoBitImage('00000:01010:00000:10001:01110:')
up = original.shift_up(1)
down = original.shift_down(1)
left = original.shift_left(1)
right = original.shift_right(1)
```

原始變數，向上，向下，左側和右側將顯示在 LED 顯示幕上，如下所示。您可以看到每個圖像都已向上、向下、向左或向右移動了一行。

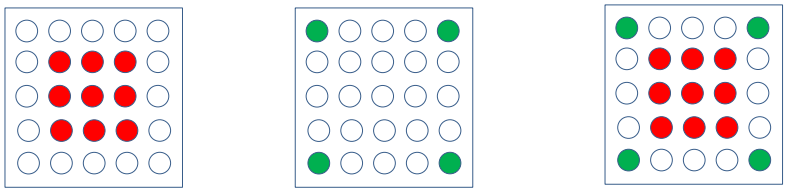


原始　向上　下　左　右

使用 + 運算子可以將多個圖像合併為一個圖像。

```
from pystubit.dsply import StuduinoBitImage
img1 = StuduinoBitImage('00000:01110:01110:01110:00000',color=(10,0,0))
img2 = StuduinoBitImage('10001:00000:00000:00000:10001',color=(0,10,0))
img3 = img 1 + img2
```

變數 img1、img2 和 img3 將顯示在 LED 顯示幕上，如下所示。您可以看到 img1 和 img2 是如何組合成 img3 的。



img1　img2　img3

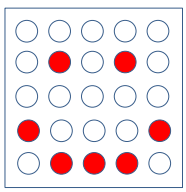
您可以使用 copy（） 方法製作圖像的副本。

4.2.6. 包含的圖像

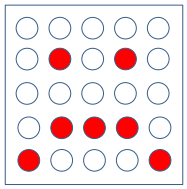
StuduinoBitImage 允許您使用預設情況下包含在 Studuino：bit 中的各種模式。

```
from pystubit.dsply import StuduinoBitImage
img1 = StuduinoBitImage.HAPPY
img2 = StuduinoBitImage.SAD
img3 = StuduinoBitImage.ANGRY
```

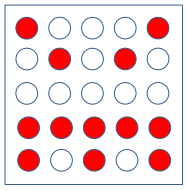
變數 img1、img2 和 img3 將顯示在 LED 顯示幕上，如下所示。



img1



img2



img3

有關使用 StuduinoBitImage 可存取的所有包含圖像的清單，請參閱第 5.3

4.3. StuduinoBitBuzzer 類

StuduinoBitBuzzer 類用於控制蜂鳴器。

4.3.1. 構造函數

使用它來創建一個操作蜂鳴器的物件。

```
from pystubit.bzr import StuduinoBitBuzzer
buzzer = StuduinoBitBuzzer()
```

4.3.2. 調整聲音

您可以使用 on（聲音， *， 持續時間=沒有） 方法從蜂鳴機播放聲音。使用「聲音」參數，通過 MIDI 音符編號或音符名稱來設置聲音的頻率。有關可用註釋編號和名稱的清單，請參見第 5.2

```
from pystubit.bzr import StuduinoBitBuzzer
buzzer = StuduinoBitBuzzer()
buzzer.on('C4', duration=1000)
```

這將使蜂鳴機播放音符 C4 1 秒鐘。

4.3.3. 發佈 PWM

Studuino：bit 使用 PWM 作為其聲音輸出。它最多可以同時使用四個 PWM。如果沒有可用的 PWM，您將無法使用您製作的任何蜂鳴器物件。您可以通過使用 release（） 方法釋放分配給不同蜂鳴器物件的 PWM 來解決此問題。

4.4. StuduinoBitButton 類

StuduinoBitButton 類用於操作 Studuino：bit Core Unit 表面上的兩個按鈕。

4.4.1. 構造函數

使用它來創建操作按鈕的物件。在參數中設置 A 或 B。

```
from pystubit.button import StuduinoBitButton
button_a = StuduinoBitButton('A')
```

4.4.2. 查找按鈕狀態

使用 get_value（） 方法尋找按鈕作為整數的值 （0/1），使用 is_pressed（） 方法將傳回真/假以告訴您是否按下按鈕。 was_pressed（） 將返回真/假，告訴您過去是否按下過某個按鈕。 get_presses（） 方法查找過去按下按鈕的次數。

```
from pystubit.button import StuduinoBitButton
button_a = StuduinoBitButton('A')
button_b = StuduinoBitButton('B')
print('Press A button')
while not button_a.is_pressed:
    pass
print('A button is pressed')
```

當按下 A 按鈕時，這將在標準顯示幕上顯示「按下按鈕」字樣。

4.5. StuduinoBitLightSensor 類

StuduinoBitLightSensor 類用於操作核心單元中的光感測器。

4.5.1. 構造函數

用它來製作一個操作光感測器的物件。

```
from pystubit.sensor import StuduinoBitLightSensor
light_sensor = StuduinoBitLightSensor()
```

4.5.2. 查找亮度

使用 get_ value（）方法查找光感測器的模擬值。它將被格式化為從 0 到 4095 的整數。

```
from pystubit.sensor import StuduinoBitLightSensor
import time
light_sensor = StuduinoBitLightSensor()
while True:
    print(light_sensor.get_value())
    time.sleep_ms(500)
```

這將每 500 毫秒在標準顯示幕上顯示一次光感測器的值。

4.6. StuduinoBitTemperature 類

StuduinoBitTemperature 類用於操作核心單元中的溫度感測器。

4.6.1. 構造函數

用它來製作一個操作溫度感測器的物件。

```
from pystubit.sensor import StuduinoBitTemperature
temperature = StuduinoBitTemperature ()
```

4.6.2. 查找溫度

使用 get_ value（）方法查找溫度感測器的模擬值。它將被格式化為從 0 到 4095 的整數。使用 get_celsius（ndigits= 2）的方法從溫度感測器獲得以攝氏度為單位的溫度。ndigits 參數可用於設置小數位數。

```
from pystubit.sensor import StuduinoBitTemperature
import time
temperature = StuduinoBitTemperature ()
while True:
    print(temperature.get_value())
    print(temperature.get_celsius())
    time.sleep_ms(500)
```

這將每 500 毫秒在標準顯示幕上顯示溫度感測器的類比值和溫度（以攝氏度為單位）。

4.7. StduinoBitAccelerometer 類

StduinoBitAccelerometer 類用於操作核心單元中的加速度計。

4.7.1. 構造函數

使用此選項可創建操作加速計的物件。

```
from pystubit.sensor import StduinoBitAccelerometer
acc = StduinoBitAccelerometer(fs='2g', sf='ms2')
```

您可以使用 StduinoBitAccelerometer 構造函數中的 fs（滿量程，即測量刻度的最大端）和 sf 參數來設置單位。在 fs 參數中設置 2g、4g、8g 或 16g，以測量高達 2g、4g、8g 或 16g 的加速度。默認情況下，它將設置為 2g。在 sf 參數中設置 ms2 或 mg，以 m/sec^2 或毫 G 為單位測量加速度。默認情況下，它將設置為 m/sec^2。

4.7.2. 加速度計設置

使用 set_fs（值）方法設置陀螺儀的最大測量值。您可以將值參數設置為 2g、4g、8g 或 16g。您還可以使用 set_sf（值）方法設置度量單位。可以設置為 ms2 或 mg。

4.7.3. 尋找加速度

使用 get_values（ndigits=2） 方法查找加速計的值。這些值將以元組 （x， y， z） 格式返回。 get_x（ndigits=2）、 get_y（ndigits=2）和 get_z（ndigits=2）方法分別找到 x 軸、y 軸和 z 軸加速度。無論使用何種方法， ndigits 參數都可用於設置小數位數。

```
from pystubit.sensor import StduinoBitAccelerometer
import time
acc = StduinoBitAccelerometer()
while True:
    print(acc.get_values())
    time.sleep_ms(500)
```

這將每 500 毫秒在標準顯示器上顯示一次加速度計值。

4.8. StuduinoBitGyro 類

StuduinoBitGyro 類用於操作核心單元中的陀螺儀。

4.8.1. 構造函數

用它來製作一個操作陀螺儀的物件。

```
from pystubit.sensor import StuduinoBitGyro
gyro = StuduinoBitGyro (fs='250dps', sf='dps'))
```

您可以使用 StuduinoBitGyro 構造函數中的 fs(測量刻度的最大端)和 sf 參數來設置單位。在 fs 參數中設置 250 輸出/秒、500 輸出/秒、1000 輸出/秒 或 2000 輸出/秒,以測量高達 250 度/秒、500 度/秒、1000 度/秒或 2000 度/秒的角速度。默認情況下，它將設置為 250 度/秒。在 sf 參數中設置每秒輸出或每秒鐘轉數,以 度/秒 或 弧度/秒 進行測量。默認情況下，它將設置為 度/秒。

4.8.2. 陀螺儀設置

使用 set_fs（值）方法設置陀螺儀的最大測量值。您可以將值參數設置為 250 輸出/秒、500 輸出/秒、1000 輸出/秒 或 2000 輸出/秒。您還可以使用 set_sf（值）方法設置度量單位。可以設置為每秒輸出或每秒鐘轉數。

4.8.3. 查找角速度

使用方法 get_values（ndigits=2）來查找陀螺儀的值。這些值將以 構造函數或 set_sf 方法指定的單位以元組 （x， y， z） 格式返回。

get_x（ndigits=2）、get_y（ndigits=2）和 get_z（ndigits=2）方法分別找到 x 軸、y 軸和 z 軸角速度。無論使用何種方法，ndigits 參數都可用於設置小數位數。

```
from pystubit.sensor import StuduinoBitGyro
import time
gyro = StuduinoBitGyro ()
while True:
    print(gyro.get_values())
    time.sleep_ms(500)
```

這將每 500 毫秒在標準顯示幕上顯示陀螺儀的值。

4.9. StuduinoBitCompass 類

StuduinoBitCompass 類用於操作核心單元中的地磁感測器（或 Compass）。

4.9.1. 構造 函數

用它來製作一個操作指南針的物件。

```
from pystubit.sensor import StuduinoBitCompass
compass = StuduinoBitCompass ()
```

4.9.2. 校準

使用 calibrate（） 方法來校准指南針。校準過程將使用 LED 顯示幕。

使用 is_ calibrated（） 方法找出指南針是否已校準。使用 clear_calibration（） 方法清除任何現有的校準數據。

4.9.3. 查找路線

使用 heading（） 方法查找方向資訊（以 0-360 度為單位）。如果您的指南針尚未校準，那麼當您運行 heading（） 方法時，將自動開始校準。0° 定義為將核心單元放置在 LED 顯示幕朝上且 USB 連接器朝南放置時。角度從那裡以順時針方式增加。

4.9.4. 查找指南針值

使用 get_ values（）方法查找指南針的值。這些值將使用 μT（微特斯拉）單位以元組（x，y，z）格式返回。

get_x（）、get_y（） 和 get_z（） 方法都可以使用，分別查找 x 軸、y 軸和 z 軸分別朝向的方向。

```
from pystubit.sensor import StuduinoBitCompass
import time
compass = StuduinoBitCompass ()
while True:
    print(compass.get_values())
    time.sleep_ms(500)
```

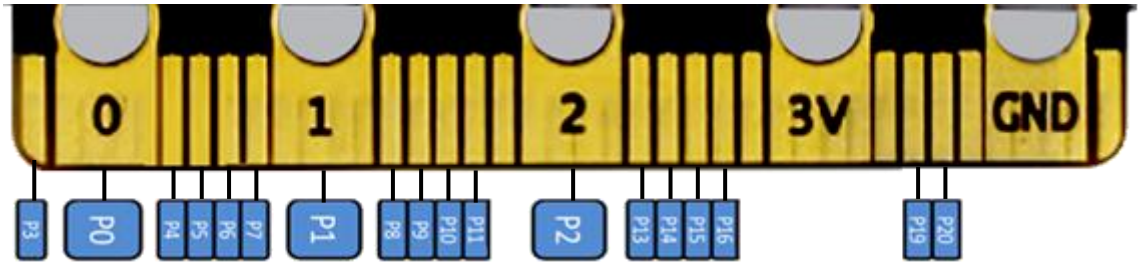
這將每 500 毫秒在標準顯示幕上顯示指南針的值。

4.10. StuduinoBitTerminal 類

StuduinoBit 終端類用於操作核心單元中的埠。

4.10.1. 構造函數

StuduinoBit 終端構造函數可用於創建從 P0 到 P20 的核心單元上所有埠的物件，P17 和 P18 除外。可與 StuduinoBit 終端類一起操作的埠如下所示，P0-P20。



使用此選項可創建一個在 P0 和 P20 之間操作指定埠的物件。

```
from pystubit.terminal import StuduinoBitTerminal
p0 = StuduinoBitTerminal('P0')
```

下面列出了 P0 和 P20 之間埠的輸入/輸出功能。

埠	輸入/輸出類型	埠	輸入/輸出類型
P0	數位輸入/輸出和模擬輸入	P10	數位輸入/輸出
P1	數位輸入/輸出和模擬輸入	p11	數位輸入/輸出
P2	數位和模擬輸入	P12	數位輸入/輸出
P3	數位和模擬輸入	P13	數位輸入/輸出
P4	數位輸入/輸出	P14	數位輸入/輸出
P5	數位輸入/輸出	P15	數位輸入/輸出
P6	數位輸入/輸出	P16	數位輸入/輸出
P7	數位輸入/輸出	P19	數位輸入/輸出
P8	數位輸入/輸出	P20	數位輸入/輸出
P9	數位輸入/輸出		

4.10.2. 數位輸入/輸出

如果您創建的埠對像是數位輸入/輸出類型，則可以使用 write_digital（值）方法發送數字信號。值參數可以設置為 0 或 1。您還可以使用 read_digital（） 方法來讀取數字信號。使用 set_analog_hz（）方法選取要使用的 PWM 輸出頻率，然後使用 write_analog（值）方法發送 PWM 信號。

4.10.3. 數位輸入/輸出和模擬輸入

如果您創建的埠對像是數位輸入/輸出和類比輸入類型，則可以使用上述數位輸入/輸出方法以及 read_analog（）方法來讀取模擬信號。

4.10.4. 數位和模擬輸入

如果您創建的埠對像是數位和類比輸入類型，則只能使用 read_digital（） 和 read_analog（） 方法。

5. 附錄

5.1. 板模組

板模組包含用於操作 Studuino：bit 硬體的物件的定義。如圖所示進行聲明以使用物件。

```
from pystubit.board import *
```

下面列出了電路板模組定義的物件。

物件	硬體
顯示	液晶顯示幕
影像（類）	液晶顯示幕（影像）
蜂鳴器	蜂鳴器
button_a， button_b	A/B 按鈕
光感測器	光感測器
溫度	溫度感測器
加速度計	加速度計
陀螺	陀螺儀
指南針	地磁感測器（指南針）
P0-16， P19， P20	邊緣連接器
引腳 0-引腳 3	邊緣連接器的 micro:bit （★）

★ :以下物件不能同時使用:引腳 0 和 p0、引腳 1 和 p1、引腳 2 和 p2、引腳 3 和 p3。

★ ：物件 pin0-pin3 可以使用與物件 p0-p3 相同的方法，但 read_analog（） 方法將返回從 0 到 1023 的值。

5.2. 聲音輸出

StuduinoBitBuzzer 類中使用的 MIDI 音符編號和音符名稱如下所示。

MIDI	Note	MIDI	Note	MIDI	Note	MIDI	Note	MIDI	Note	MIDI	Note	MIDI	Note
48	C3	60	C4	72	C5	84	C6	96	C7	108	C8	120	C9
49	CS3	61	CS4	73	CS5	85	CS6	97	CS7	109	CS8	121	CS9
50	D3	62	D4	74	D5	86	D6	98	D7	110	D8	122	D9
51	DS3	63	DS4	75	DS5	87	DS6	99	DS7	111	DS8	123	DS9
52	E3	64	E4	76	E5	88	E6	100	E7	112	E8	124	E9
53	F3	65	F4	77	F5	89	F6	101	F7	113	F8	125	F9
54	FS3	66	FS4	78	FS5	90	FS6	102	FS7	114	FS8	126	FS9
55	G3	67	G4	79	G5	91	G6	103	G7	115	G8	127	G9
56	GS3	68	GS4	80	GS5	92	GS6	104	GS7	116	GS8		
57	A3	69	A4	81	A5	93	A6	105	A7	117	A8		
58	AS3	70	AS4	82	AS5	94	AS6	106	AS7	118	AS8		
59	B3	71	B4	83	B5	95	B6	107	B7	119	B8		

5.3. 包含的圖像

Studuino：bit 庫中包含的圖像如下所示。

StuduinoBitImage.HEART
StuduinoBitImage.HEART_SMALL
StuduinoBitImage.HAPPY
StuduinoBitImage.SMILE
StuduinoBitImage.SAD
StuduinoBitImage.CONFUSED
StuduinoBitImage.ANGRY
StuduinoBitImage.ASLEEP
StuduinoBitImage.Surprise
StuduinoBitImage.SILLY
StuduinoBitImage.FABULOUS
StuduinoBitImage.MEH
StuduinoBitImage.YES
StuduinoBitImage.NO
StuduinoBitImage.CLOCK12,
StuduinoBitImage.CLOCK11,
StuduinoBitImage.CLOCK10,
StuduinoBitImage.CLOCK9,
StuduinoBitImage.CLOCK8,
StuduinoBitImage.CLOCK7,
StuduinoBitImage.CLOCK6,
StuduinoBitImage.CLOCK5,
StuduinoBitImage.CLOCK4,
StuduinoBitImage.CLOCK3,
StuduinoBitImage.CLOCK2,
StuduinoBitImage.CLOCK1
StuduinoBitImage.ARROW_N,
StuduinoBitImage.ARROW_NE,
StuduinoBitImage.ARROW_E,
StuduinoBitImage.ARROW_SE,
StuduinoBitImage.ARROW_S,
StuduinoBitImage.ARROW_SW,
StuduinoBitImage.ARROW_W,
StuduinoBitImage.ARROW_NW
StuduinoBitImage.TRIANGLE
StuduinoBitImage.TRIANGLE_LEFT
StuduinoBitImage.CHESSBOARD
StuduinoBitImage.DIAMOND

StuduinoBitImage.DIAMOND_SMALL
StuduinoBitImage.SQUARE
StuduinoBitImage.SQUARE_SMALL
StuduinoBitImage.RABBIT
StuduinoBitImage.COW
StuduinoBitImage.MUSIC_CROTCHET
StuduinoBitImage.MUSIC_QUAVER
StuduinoBitImage.MUSIC_QUAVERS
StuduinoBitImage.PITCHFORK
StuduinoBitImage.XMAS
StuduinoBitImage.PACMAN
StuduinoBitImage.TARGET
StuduinoBitImage.TSHIRT
StuduinoBitImage.ROLLERSKATE
StuduinoBitImage.DUCK
StuduinoBitImage.HOUSE
StuduinoBitImage.TORTOISE
StuduinoBitImage.BUTTERFLY
StuduinoBitImage.STICKFIGURE
StuduinoBitImage.GHOST
StuduinoBitImage.SWORD
StuduinoBitImage.GIRAFFE
StuduinoBitImage.SKULL
StuduinoBitImage.UMBRELLA
StuduinoBitImage.SNAKE

5.4. 類表

此表列出了可用於每個類的方法，以及這些類對應於板模組中的哪些物件。

功能	類	板模組	方法	指示
按鈕	StuduinoBitButton	button_a button_b	__init__（ab）	用於製作操作 A/B 按鈕的物件。ab 參數可以設置為 A 或 B。
			get_value（）	如果按下按鈕，則返回 0；如果未按下按鈕，則返回 1。
			is_pressed（）	如果按下按鈕，則返回真。
			was_pressed（）	按鈕物件存儲按鈕已按下的資訊。如果過去曾按下按鈕，則此方法返回真。調用此方法時，將重置資訊。
			get_presses（）	按鈕物件存儲按鈕已按下的資訊。此方法返回過去按下按鈕的次數。調用此方法時，計數將重置。
液晶顯示幕（全彩）	StuduinoBitDisplay	顯示	__init__（）	製作顯示物件。
			get_pixel（x， y）	返回 y 列 x 行處 LED 的顏色。顏色將顯示在（R，G，B）中。
			set_pixel（x， y，顏色）	將 LED 的顏色設定為 y 列 x 行。可以使用（R，G，B），[R，G，B] 或 #RGB 設置參數。
			clear（）	將顯示器中所有 LED 的亮度設置為 0（熄滅）。
			show（可迭代，延遲=400， *， 等待=真，循環=假，清除=假， 顏色=沒有）	按順序顯示可反覆運算參數（圖像、字串或數位）。
			scroll（字串， 延遲=150， *， 等待	在顯示幕上水平滾動值參數（字母/數位）。

			=真，循環=假， 顏色=無）	
			on（）	打開 LED 顯示螢幕的電源。
			off（）	關閉 LED 顯示螢幕的電源。(這允許您將連接到顯示器的 GPIO 終端用於其他目的。
			is_on（）	如果顯示器的電源為「開」，則返回真;如果顯示器的電源為“關”，則返回假。
圖像	StduinoBitImage	圖像	__init__（字串，顏色=無）__init__（寬度=無，高度=無，緩衝區=無，顏色=無）	使用字串參數中以 0 秒（LED 關閉）和 1 秒（LED 亮）寫入的圖案製作圖像物件。 StduinoBitImage（'01100 : 10010 : 11110 : 10010 : 10010 : ', color=（0，0，10）） 使用指定寬度（寬度參數）和高度（hight 參數）內的空格創建圖像物件。 StduinoBitImage（2， 2，bytearray（[0，1，0，1]） StduinoBitImage（3， 3） 與 micro：bit 一樣，可以使用任何數位 0-9，但 1-9 都轉換為 ON，而 0 表示 OFF。如果尚未設置顏色變數，則 （RGB）= （31，0，0）。
			width（）	返回圖像在列中的寬度。
			height（）	返回圖像在行中的高度。
			set_pixel（x， y，值）	value 參數設置圖像中座標 （x， y） 處的圖元值。值參數可以設置為 0 （OFF） 或 1 （ON）。
			set_pixel_color(x，	顏色參數設定圖像中座標 （x， y） 處像素的顏色。可以使用 （R， G，

			y， 顏色）	B）、[R， G， B] 或 #RGB 来设置颜色参数。
			get_pixel（x， y）	返回影像中座標 （x， y） 處的像素值。
			get_pixel_color（x， y，hex=false）	返回影像中座標 （x， y） 處像素的顏色。 如果十六進位參數設置為 False，則顏色將寫入 （R，G，B）。如果設置為真，它將以#RGB 形式寫入。
			set_base_color（自身，彩色）	顏色參數設置圖像中所有像素的顏色。可以使用 （R， G， B）、[R，G， B] 或 #RGB 来设置颜色参数。
			shift_left（n）	通過將當前圖像移位到 n 參數中設置的剩餘空格數來生成下一個圖像。
			shift_right（n）	與 shift_left（-n） 相同。
			shift_up（n）	通過在 n 參數中將當前圖像移位設置的多個空格來生成下一個圖像。
			shift_down（n）	與 shift_up（-n） 相同。
			copy（）	返回圖像的完整副本。
			repr（圖像）	獲取表示圖像的緊湊字串。
			str（圖像）	獲取表示圖像的可讀字串。
			+	合併兩個圖像中的所有圖元以創建新圖像。
				包含的圖像 （請參閱 5.3）
港口	StuduinoBitTerminal	請參閱下面的埠模組。	__init__（針）	創建一個應用於指定的 Studuino：bit 埠 P0-P16、P19 或 P20 的終端物件。
數位輸入/輸出和模擬輸入（※）	StuduinoBitAnalogdigitalPin	p0， p1， p2， p3 注意：	write_digital（值）	如果值參數為 1，則將數位信號設置為「高」;如果值參數為 0，則將數位信號設置為“低”。
			read_digital（）	讀取數字信號。如果接收的值為 0，則返回 High;如果接收的值為 1，

		p2 和 p3 不能用於數位輸入/輸出。對它們使用 write_digital 方法將導致錯誤。read_digital 方法是通過模擬輸入實現的，因此可以使用。		則返回 Low。
			write_analog（值）	通過埠發送 PWM 信號。值參數可以設置為 0 （0%） 到 1023 （100%）。
			set_analog_period（週期，計時器=-1）	設置 PWM 信號的週期（以毫秒為單位）。
			set_analog_period_microseconds（週期，計時器=-1）	以微秒為單位設置 PWM 信號的週期。
			set_analog_hz（hz， 定時器=-1）	按頻率設置 PWM 信號的週期。
			狀態（）	顯示 PMM 的當前使用狀態。
			read_analog（mv=False）	從埠讀取電壓，如果 mv 參數設置為 False，則將其作為介於 0 （0V）和 4096 （3.3V） 之間的整數返回。如果 mv=True，則電壓將以 mV 為單位寫入。
數位輸入/輸出（★）	StuduinoBitDigitalPin	p4-p16， p19，p20	write_digital（值）	如果值參數為 1，則將數位信號設置為「高」;如果值參數為 0，則將數位信號設置為“低”。
			read_digital（）	讀取數字信號。如果接收的值為 0，則返回 High;如果接收的值為 1，則返回 Low。
			write_analog（值）	通過埠發送 PWM 信號。值參數可以設置為 0 （0%） 到 1023 （100%）。
			set_analog_period（週期，計時器	設置 PWM 信號的週期（以毫秒為單位）。

			=-1）	
			set_analog_period_microseconds（週期，計時器=-1）	以微秒為單位設置 PWM 信號的週期。
			set_analog_hz（hz，定時器=-1）	按頻率設置 PWM 信號的週期。
			狀態（）	顯示 PMM 的當前使用狀態。
蜂鳴器	StduinoBitBuzzer	蜂鳴器	__init__（）	製作操作蜂鳴器的物件。
			on（聲音，*，持續時間=-1）	在聲音和持續時間參數中設置蜂鳴器的聲音。聲音可以使用音符名稱（C3-G9），MIDI 音符編號（48-127）或頻率（作為整數）來設置，持續時間可以以毫秒為單位進行設置。如果未指定持續時間參數，蜂鳴機將繼續播放，直到您調用 off 方法。 bzr.on（'50'，Duration=1000）# 蜂鳴器將播放對應於 MIDI 音符編號 50 的音符 1 秒。 bzr.on（'C4'，Duration=1000）# 蜂鳴器將播放音符 C4 1 秒 bzr.on（440）
			off（）	使蜂鳴器停止播放聲音。
溫度感測器	StduinoBitTemperature	溫度	__init__（）	製作操作溫度感測器的物件。
			get_value（）	返回核心單元中溫度感測器的值 （0-4095）。
			get_celsius（）	返回核心單元中溫度感測器的值（以攝氏度為單位）。
光感測器	StduinoBitLightSensor	光感測器	__init__（）	製作操作光感測器的物件。
			get_value（）	返回核心單元中光感測器的值 （0-4095）。
加速度 計	StduinoBitAccelerometer	加速度計	__init__（fs='2G'，	通過將 fs 參數設置為 2g、4g、8g 或 16g 以設置要測量的最大加速

			sf='mg2'）	度，並將 sf 參數設置為 mg 或 ms2 來選取度量單位，從而創建加速計物件。 默認設置為 fs=2G 和 sf=ms2。
			get_x	沿 x 軸測量加速度。
			get_y	沿 y 軸測量加速度。
			get_z	沿 x 軸測量加速度。
			get_values（）	=（get_x（）， get_y（）， get_z（））
			set_axis（模式）	模式可以使用以下字串之一進行設置：sbmp（對於 Studuino：bit MicroPython），sbs（對於 Studuino：bit 軟體）或 mb（對於 micro：bit）。 默認情況下，它將設置為 sbmp。
			set_fs（值）	設置要測量的最大加速度為 2g、4g、8g 或 16g。
			set_sf（值）	使用 mg 或 ms2 設置度量單位。
陀螺儀	StuduinoBitGyro	陀螺	__init__（fs='250dps'，sf='dps'）	通過將 fs 參數設置為 250dps、500dps、1000dps 或 2000dps 以設置要測量的最大角速度，並將 sf 參數設置為 dps 或 rps 來選取測量單位，從而製作陀螺儀物件。 默認設置為 fs=250dps 和 sf=dps。
			get_x	沿 x 軸測量角速度。
			get_y	沿 y 軸測量角速度。
			get_z	沿 x 軸測量角速度。
			get_values（）	=（get_x（）， get_y（）， get_z（））
			set_axis（模式）	模式可以使用以下字串之一進行設置：sbmp（對於 Studuino：bit MicroPython），sbs（對於 Studuino：bit 軟體）或 mb（對於 micro：bit）。

				默認情況下，它將設置為 sbmp。
			set_fs（值）	設置要以 250dps、500dps、1000dps 或 2000dps 測量的最大角速度。
			set_sf（值）	使用 dps 或 rps 設置度量單位。
指南針	StuduinoBitCompass	指南針	__init__（）	製作操作指南針的物件。最大可測量值為±4500μT，單位為 μT（微特斯拉）。
			get_x	沿 x 軸查找磁力。
			get_y	沿 y 軸查找磁力。
			get_z	沿 z 軸查找磁力。
			get_values（）	=（get_x（）， get_y（）， get_z（））
			set_axis（模式）	模式可以使用以下字串之一進行設置：sbmp（對於 Studuino：bit MicroPython），sbs（對於 Studuino：bit 軟體）或 mb（對於 micro：bit）。默認情況下，它將設置為 sbmp。
			calibrate（）	啟動指南針校準過程。旋轉 核心單元 ，直到 LED 顯示幕在其邊框周圍繪製一個完整的圓圈。
			is_calibrated（）	如果指南針已校準，則返回 false True;如果尚未校準，則返回 False。
			clear_calibration（）	擦除現有校準數據。
			heading（）	查找指南針朝向的方向。整數為 0-360°，順時針計數，180°為北。
斷續器	StuduinoBitBLE			T.B.D
無線通信	StuduinoBitRadio	此類沒有物件。	__init（）__	製作用於操作設備無線通信的物件。
			on（）	打開無線通信。必須顯式調用無線通信，因為它們會消耗功率並佔用其他功能可能需要的記憶體。

			off（）	關閉無線通信，節省設備的電源和記憶體。
			除非已打開無線通信，否則無法使用以下方法。	
			start（組）	將組參數設置為 0 到 255 之間的數位，以選取組並開始無線通信。
			send_number（n）	廣播一個數位。該數位將是一個 32 位整數。
			send_value	廣播數位和變數名稱。該數位將是一個 32 位整數，變數名稱的長度最多為 13 個字元。
			send_string	廣播字串。字串的長度最多為 19 個字元。
			send_buffer	廣播位元組序列。它最多可以有 19 個字節。
			recv（）	接收數據。如果未收到任何數據，則返回 None。
			group（組=-1）	將組參數設置為介於 0 和 255 之間的數位以選取組。
無線網路	StuduinoBitWiFi			T.B.D

（★） 各個數位輸入/輸出、模擬輸入以及數位和類比輸入/輸出類旨在製作成實例。實例旨在通過終端類創建。

082683 K0302