Traducción de la API de Function Extension de mBot2

[**Display 2**](#_gtra8sly4swf)

[**Sprites 2**](#_6esmku7scja4)

[Configuración del fondo 2](#_ze8adtazgbyp)

[Configurando el fondo. 2](#_u2yjdnsbeav)

[Creando Sprites 2](#_uhprxrpuforw)

[Moving Sprites 4](#_rcwzeolmixye)

[Rotating Sprites 5](#_vbpchu7b5inb)

[Scaling Sprits 5](#_skoocdk8muk)

[Setting Colors 6](#_6wg0mmbncl5m)

[Layer and display control 6](#_ae8zlhtu455q)

[Rendering 8](#_rw10pgc9elgw)

[Doodle 9](#_vx2r1vqqbpf)

link del articulo

<https://education.makeblock.com/help/mblock-python-editor-apis-for-function-extension/>

ultima actualización: Last Updated OnDecember 22, 2021

raiz del articulo:

* [Help Hub](https://education.makeblock.com/help/)
* [mBlock Python Editor](https://education.makeblock.com/help/category/mblock-python/)
* [Python API Documentation for Devices](https://education.makeblock.com/help/category/mblock-python/mblock-python-editor-python-api-documentation-for-devices/)
* [Python API Documentation for CyberPi](https://education.makeblock.com/help/category/mblock-python/mblock-python-editor-python-api-documentation-for-devices/mblock-python-editor-python-api-documentation-for-cyberpi/)
* APIs for Function Extension

Para ayudarlo a usar las API de manera más eficiente, los escenarios de aplicación de las API están etiquetados de la siguiente manera:

: indica que una API es compatible con la programación de Python 3 y se puede usar en modo en vivo en el editor mBlock Python.

: indica que una API es compatible con la programación de MicroPython y se puede usar en el modo de carga en el editor mBlock Python.

: indica que una API es compatible con la programación de Python 3 y MicroPython y se puede usar en los modos Live y Upload en el editor mBlock Python.

# Display

Además de extender funciones a través de placas de extensión y módulos electrónicos, CyberPi puede extender funciones a través de sus componentes y partes. Actualmente, las API de Sprites y Doodle se proporcionan para ampliar las funciones de la pantalla de CyberPi.

**Observacion**: Los metodos de las API de Sprites y Doodle solo admiten programación en MicroPython y no se pueden usar en programación con Python 3. Además, use estas API solo en el modo de carga. Para obtener detalles sobre la descripción y las instrucciones de los modos de programación, consulte “ Modos Live y Upload ”.

# Sprites

Las API de Sprites le permiten aprovechar al máximo la pantalla de CyberPi para creación. de juegos y IU.

## Configuración del fondo

**cyberpi.background.fill(r,*g*,*b* )**

Este metodo permite crear un fondo de pantalla de un solo color uniforme. No permite fondo multicolor.

Parámetros:

r: int , intensidad del color rojo, que va de 0 a 255

g: int , intensidad del color verde, que va de 0 a 255

b: int , intensidad del color azul, que va de 0 a 255

**cyberpi.background.copy()**

Este método permite copiar los elementos presentes en pantalla para mostrarlos como fondo.

Actualmente, esta API no está abierta a los usuarios.

## Creando Sprites

**sprite1 = cyberpi.sprite()**

Este método permite crear un sprite

La siguiente tabla describe los atributos predeterminados de un sprite cuando se crea.

**Atributo Valor por defecto**

coordenada x: 64

coordenada y: 64

Dirección: 90° (hacia la derecha)

Tamaño: 100%

punto de anclaje: Centro del sprite

Desplegado: Sí

Color: Blanco

Capa: El último creado se encuentra en la capa superior.

**sprite.draw\_text(*text*)**

Este método permite hacer un Sprite de texto con un tamaño de letra de 16 píxeles por defecto.

Parámetro:

***text***: str, representa la cadena de caracteres que se usará como sprite.

Si ingresa un texto en un idioma que no es compatible con la biblioteca de caracteres, se muestran caracteres basura. CyberPi admite los siguientes idiomas:

* Chino simplificado ( SC)
* Inglés ( ES)
* francés | francés
* español | inglés
* Chino tradicional | 繁體中文
* alemán | Alemán
* italiano | italiano
* portugués | portugués
* ruso | русский язык
* coreano | 한국어
* Japonés |日本語
* holandés | Holanda

**sprite.draw\_pixel(*pixel*, *x\_size*, *y\_size*)**

Este método permite crear un Sprite en base a una matriz de píxeles.

Parámetros:

***pixel***: array, representa la matriz hexadecimal con colores de los píxeles de izquierda a derecha y de arriba a abajo del patrón.

Por ejemplo, [0xFF0000, 0x00FF00, 0x0000FF].

***x\_size***: int, ancho del patrón de píxeles, por defecto 16

***y\_size***: int, altura del patrón de píxeles, por defecto 16

Los colores establecidos para los píxeles que exceden el ancho y la altura definidos por los parámetros se descartan y las áreas restantes se definen transparentes.

**sprite.draw\_QR(*content*)**

Este método permite crear un código QR como Sprite.

Parámetro:

*contenido*: str, contenido del código QR que se usará como sprite

Ejemplo

importar cyberpi

2

3 qrcode = cyberpi.sprite()

4 qrcode.draw\_qrcode("Bienvenido")

5 qrcode.set\_size(tamaño=300)

6 qrcode.show()

7 cyberpi.screen.render()

**sprite.draw\_sketch()**

Este método permite definir un boceto como un sprite

**sprite.mirror(*axis*)**

Este método permite reflejar vertical u horizontalmente.

Parámetro:

***axis***: str, el rango de valores,"x": reflejo vertical y “y": reflejo horizontal

**sprite.delete()**

Este método permite eliminar un Sprite

## Moving Sprites

**sprite.set\_align(*align\_point* = "center")**

Este método permite definir el punto de anclaje del sprite. Este determina el centro de movimiento y giro.

Parámetro:

***align\_point***: str , representa la posición del punto de anclaje;

rango de valores:

* "top\_mid": en el centro superior del sprite
* "top\_left": en la parte superior izquierda del sprite
* "top\_right": en la parte superior derecha del sprite
* "center": en el centro del sprite
* "mid\_left": en el medio a la izquierda del sprite
* "mid\_right": en el medio a la derecha del sprite
* "bottom\_mid": en el centro inferior del sprite
* "bottom\_left": en la parte inferior izquierda del sprite
* "bottom\_right": en la parte inferior derecha del sprite

**sprite.move\_x(*step*)**

Este método permite mover el Sprite a lo largo del eje x.

Parámetro:

***paso***: int , representa el número de píxeles que se moverá el sprite

**sprite.move\_y(*step*)**

Este método permite mover el sprite a lo largo del eje y

Parámetro:

***paso***: int , número de píxeles que se moverá el sprite

**sprite.move\_to(*x*,*y*)**

Este método permite mover el Sprite a la posición especificada en coordenadas.

Parámetros:

***x***: int , coordenada x de la posición

***y***: int , coordenada y de la posición

**sprite.move\_random()**

Este método permite mover el sprite a una posición aleatoria.

Esta API establece las coordenadas x e y de la posición aleatoriamente en valores que van de 0 a 127.

## Rotating Sprites

**sprite.rotate(*angle*)**

Este método permite girar el sprite en el sentido de las agujas del reloj según los grados especificados.

Parámetro:

ángulo: int

**sprite.rotate\_to(*angle* = 90)**

Este método permite girar el sprite en la dirección especificada en grados

Parámetro:

***ángulo***: int , 90° por defecto

## Scaling Sprits

**sprite.set\_size(*size* = 100)**

Este método permite escalar un Sprite.

Parámetro:

***tamaño***: int , 100% por defecto

**sprite.set\_scale(*x\_size* = 100, *y\_size* = 100)**

Este método permite escalar un Sprite por porcentaje de manera independiente ancho y alto.

Parámetros:

***x\_size***: int , porcentaje para escalar el ancho del sprite, 100% por defecto

***y\_size***: int , porcentaje para escalar la altura del sprite, 100% por defecto

Actualmente, esta API no está abierta a los usuarios.

## Setting Colors

**sprite.set\_color(*r*, *g*, *b*)**

Este método permite asignar un al sprite.

Con esta API, todos los píxeles que no están vacíos se representan en el color especificado.

Parámetros:

***r***: int , intensidad del color rojo del filtro de color, que va de 0 a 255

***g***: int , intensidad del color verde del filtro de color, que va de 0 a 255

***b***: int , intensidad del color azul del filtro de color, que va de 0 a 255

**sprite.close\_color()**

Este metodo permite eliminar el filtro de color del metodo set\_color().

## **Layer and display control**

**sprite.show()**

Este método permite mostrar un Sprite

**sprite.hide()**

Este método permite mostrar un Sprite

**sprite.z\_up(*paso*)**

Este método permite mover entre capas un Sprite.

Parámetro:

***paso***: int , representa el número de capas por las que se adelanta el sprite

**sprite.z\_down(*paso*)**

Este método mueve hacia atrás un número indicado de capas.

Parámetro:

***pas***t***o***: int , representa el número de capas por las que se devuelve el sprite

**sprite.z\_max()**

Este método permite mover un Sprite a la primera capa.

**sprite.z\_min()**

Este método permite mover un Sprite a la última capa.

**screen.color\_get(***x***,** *y***)**

Este método permite obtener el color del píxel de pantalla indicado por coordenadas.

Parámetros:

***x***: int , coordenada x del píxel, que va de 0 a 127

***y***: int , coordenada y del píxel, que va de 0 a 127

**retorno**

Este método devuelve un dato tipo str que representa el color del pixel en formato hexadecimal

**sprite.is\_touch(*other\_sprite*)**

Este método determina si el Sprite está tocando a otro, o si está en contacto con el borde.

Se considera que un Sprite toca a otro cuando este ocupa, con sus pixeles no vacios, una posición que los pixeles no vacíos de otro Sprite también ocupan.

Se considera que un Sprite toca el borde de la pantalla cuando alguno de sus pixeles tiene una posición igual o mayor a 128

Parámetro:

***otro\_sprite***: sprite o str; si es str, solo puede tomar “edge”

**retorno**

Este método retorna un dato tipo bool

**sprite.get\_align()**

Este método permite consultar el punto de anclaje del sprite.

**retorno**

Este método retorna un dato tipo string que representa cualitativamente la posición del anclaje

rango de valores:

* "top mid": en el centro superior del sprite
* "top\_left": en la parte superior izquierda del sprite
* "top\_right": en la parte superior derecha del sprite
* "center": en el centro del sprite
* "mid\_left": en el medio a la izquierda del sprite
* "mid\_right": en el medio a la derecha del sprite
* "bottom\_mid": en el centro inferior del sprite
* "bottom\_left": en la parte inferior izquierda del sprite
* "bottom\_right": en la parte inferior derecha del sprite

**sprite.get\_x()**

Este método permite consultar la posición en horizontal del centro del Sprite.

**retorno**

Este método devuelve un dato tipo int

**sprite.get\_y()**

Este método permite consultar la posición vertical del centro del Sprite.

**Retorno**

Este método devuelve un dato tipo int

**sprite.get\_rotation()**

Este método permite consultar el ángulo de rotación del Sprite.

Retorno

Este método retorna un dato tipo int entre -180 a 180 grados, representa la orientación del Sprite

**sprite.get\_xsize()**

Este método permite consultar el ancho del Sprite.

**Retorno**

Este método retorna un dato tipo int que representa el ancho del Sprite

**sprite.get\_ysize()**

Este método permite consultar la altura del Sprite.

**Retorno**

Este método retorna un dato tipo int que representa la altura del Sprite.

## Rendering

La renderización es un proceso muy importante. Los atributos como la configuración, movimiento, tamaño y dirección de un sprite solo surten efecto después de Renderizar un Sprite.

**cyberpi.screen.enable\_autorender()**

Este método permite habilitar la renderizacion por 10 fotogramas de la pantalla

**cyberpi.screen.disable\_autorender()**

Este metodo deshabilita la representación predeterminada de la pantalla

**cyberpi.screen.render()**

Este método habilita el renderizado manual

# Doodle

El pincel para hacer Doodle tiene los siguientes valores de atributos por defecto.

Atributo Valor por defecto

coordenada x: 64

coordenada y: 64

Peso del trazo: 1 píxel

Color: Blanco

¿Dejar para empezar a garabatear? No

El pincel cuenta con los siguientes métodos.

**cyberpi.sketch.clear()**

Este método permite eliminar todos los bocetos.

**cyberpi.sketch.start()**

Este método permite activar el pincel para empezar a dibujar

**cyberpi.sketch.end()**

Este método permite desactivar el pincel

**cyberpi.sketch.set\_speed(*speed*)**

Este método permite configurar la velocidad del pincel.

Parámetro:

***velocidad***: int , velocidad a la que se mueve el aerógrafo, que va de 1 a 10

**cyberpi.sketch.set\_color(*r*, *g*, *b*)**

Este método permite asignar un color al pincel.

Parámetros:

*r: int or str;*

*r: int: representa la intensidad del color rojo de 0 a 255*

*r: str: representa el color del pincel: rango de valores:*

“red”, “r”,”orange”,”o”,”yellow”, “y”,”green”,”g”,

”cyan”,”c”,”blue”,”b”,”purple”, “p”,”white”, “w”,”black”, “k”

***g***: int , intensidad del color verde del aerógrafo, que va de 0 a 255

***b***: int , intensidad del color azul del aerógrafo, que va de 0 a 255

**cyberpi.sketch.set\_size(*size*)**

Este método determina el trazo del pincel. El pincel genera una punta redonda

Parámetro:

***tamaño***: int, representa el tamaño del trazo del pincel.

**cyberpi.sketch.cw(*angle*)**

Este método permite girar el pincel en los grados indicados.

Parámetro:

***ángulo***: int , representa el número de grados que gira el pincel

**cyberpi.sketch.ccw(*angle*)**

Este método gira el pincel en los grados indicados en contra del sentido del reloj.

Parámetro:

***ángulo***: int , representa el número de grados que gira el aerógrafo

**cyberpi.sketch.set\_angle(angle)**

Establece el ángulo de las caras del aerógrafo

**cyberpi.sketch.move(*step*)**

Este método permite avanzar el pincel la cantidad indicada de pixeles, el movimiento es según su orientación.

Parámetro:

***paso***: int, representa el número de píxeles que se mueve el pincel

**cyberpi.sketch.move\_x(*paso*)**

Este método permite mover el pincel en horizontal según la cantidad de pixeles indicados

Parámetro:

***paso***: int, representa el número de píxeles que el pincel se mueve a lo largo del eje x

**cyberpi.sketch.move\_y(*step*)**

Este método mueve el pincel verticalmente el número especificado de píxeles.

Parámetro:

***paso***: int , representa el número de píxeles que el pincel se mueve a lo largo del eje y

**cyberpi.sketch.move\_to(x, y)**

Este método permite mover el pincel a la coordenada indicada

Parámetros:

***x***: int ,representa la coordenada x del aerógrafo

***y***: int , representa la coordenada y del aerógrafo

**cyberpi.sketch.circle(*r*, *angulo*)**

Este método permite dibujar un arco circular con un radio y ángulo indicados. su centro se ubicará de manera perpendicular a la orientación del pincel.

Parámetros:

***r***: int , representa el radio del arco circular.

***y***: int , representa el ángulo del arco circular.

**cyberpi.sketch.get\_x()**

Este método permite consultar la coordenada x del pincel

.

**retorno**

Este método retorna un dato tipo int

**cyberpi.sketch.get\_y()**

Este método permite consultar la coordenada y el pincel.

**retorno:**

Este método retorna un dato tipo int.

**cyberpi.sketch.get\_size()**

Este método permite consultar el tamaño del trazo del pincel.

**Retorno:**

Este método retorna un dato tipo int

**cyberpi.sketch.get\_angle()**

Este método permite consultar la orientación del pincel.

**Retorno:**

Este método retorna un dato tipo int.

**cyberpi.sketch.if\_start()**

Este método permite detectar cuando el pincel se activa.

**Retorno:**

Este método retorna un valor bool

**sprite\_sketch = sprite.draw\_sketch()**

Este método permite convertir un boceto a Sprite. Su punto de anclaje se ubicará en la coordenada (64,64) y se presentará en el centro de la pantalla al crearlo.