APIs Para Cyberpi

Bienvenidos a aprender a través del uso de CyberPi a programar en Python.  
-Link oficial (en inglés): [APIs for CyberPi | Makeblock Education](https://education.makeblock.com/help/mblock-python-editor-python-api-documentation-for-cyberpi/)  
  
La herramienta CyberPi nos da varias funciones de entradas y salidas, permitiéndonos asíinteractuar con nuestros códigos y poder visualizarlos.  
  
El sistema CyberPi tiene bluetooth y wifi incorporados, los cuales pueden ser usados por Lan o por conexión a internet. Puede ir creando su propio programa y este documento te ayudará a comprender y a dominar el conocimiento y las habilidades relacionadas con campos variados, tal como IoT, inteligencia artificial, computación y redes.  
  
Además, CyberPi acepta más de 30 módulos electrónicos y múltiples extensiones, lo que nos permitirá crear varios proyectos, como una granja inteligente, una casa inteligente, un auto autónomo y, sobre todo, robots de competencia.  
La librería CyberPi trae todas esas funciones.  
  
Importando la Librería *cyberpi*

La librería cyberpi que describiremos tiene 2 distintas funciones en sí misma:   
 1.-Librería Micro Python, que solo funcione en cyberpi  
 2.-Librería python, que puede correr en los computadores

Ambas librerías traen las mismas APIs y casi las mismas salidas.  
  
Vienen estas dos librerías juntas a proposito, asi micro python puede compilar en el modo de carga (ejecución en cyberpi), y python compila en el modo en vivo (ejecución en el computador), siendo una buena manera de hacer un aprendizaje de micro python adecuado para transicionar a phyton.  
  
(PY): Nos indica que la API es compatible con la versión de python y puede usarse en el modo en vivo del editor Mblock  
  
: Nos indica que la APIes compatible con micro python y se puede usar en el modo carga del editor Mblock  
  
(símbolo y ):Nos indica que la API es compatible tanto para python como para micro python y se puede usar tanto en el modo carga como en el modo en vivo en el editor Mblock.  
  
  
Se puede usar la librería *cyberpi* en el editor de python o en Mblock 5   
  
*#no omitir la codificación de micropython “cyberpi”  
import cyberpi*

*cyberpi.console.print (“hello”)*

*#micropython omitiendo el código “cyberpi”  
from cyberpi import \*  
console.print(“Hello”)*

Para python es preferible la forma sin omisiones, así podemos dar las indicaciones más claras y se evitan los errores de compilado del código al llamar a otras librerías.  
  
La omisión es más factible para Micro Python (Modo de carga) que permite escribir menos palabras.  
  
Mblock 5 utiliza la transcodificación sin omisiones en la mayoría de los casos para evitar errores.  
APIs en la librería *cyberpi*  
  
La librería *cyberpi* proporciona varias APIs para controlar los hardware. este instructivo describe los APIs de cyberpi  
Para conocer Las APIs proporcionadas para las extensiones, consulte:   
  
APIs for function extension  
APIs for Extension board  
APIs for mBuild Modules   
  
APIs para Cyberpi  
  
Estas APIs se pueden usar para controlar Cyberpi, por lo que permite que cyberpi pueda realizar sus funciones.  
  
Las APIs para cyberpi incluyen lo siguiente:  
  
Audio

La Cyberpi está equipada con un parlante y con un micrófono. Puedes usar la siguiente API para poder grabar y reproducir archivos de audio, ademas de poder controlar el volumen y la velocidad de su reproducción.  
  
Archivos de audio Precargados  
  
*cyberpi.audio.play(nombre\_musica)*

Reproduce un audio precargado.

Esta API bloquea el uso hasta que termine la reproducción.  
  
Parámetro:   
Music str(String): Nombre del archivo del audio que quieres reproducir. La tabla “ Rango de ajustes y efectos de sonido”, como dice su nombre, puede regular los ajustes del sonido a reproducir.

“Rango de musica y efectos de sonido”

|  |  |  |  |
| --- | --- | --- | --- |
| **Categoría** | **Nombre** | **Nombre de la variable** | **Efecto de sonido** |
| **01-Emociones** | hello | SPEAKER.hello | hello |
| hi | SPEAKER.hi | hi |
| bye | SPEAKER.bye | bye |
| yeah | SPEAKER.yeah | yeah |
| wow | SPEAKER.wow | wow |
| laugh | SPEAKER.laugh | laugh |
| hum | SPEAKER.hum | hum |
| sad | SPEAKER.sad | sad |
| sigh | SPEAKER.sigh | sigh |
| annoyed | SPEAKER.annoyed | annoyed |
| angry | SPEAKER.angry | angry |
| surprised | SPEAKER.surprised | surprised |
| yummy | SPEAKER.yummy | yummy |
| curious | SPEAKER.curious | curious |
| embarrassed | SPEAKER.embarrassed | embarrassed |
| ready | SPEAKER.ready | ready |
| sprint | SPEAKER.sprint | sprint |
| sleepy | SPEAKER.sleepy | sleepy |
| meow | SPEAKER.meow | meow |
| **02-Sonidos electrónicos** | start | SPEAKER.start | start |
| switch | SPEAKER.switch | switch |
| beeps | SPEAKER.beeps | beeps |
| buzzing | SPEAKER.buzzing | buzzing |
| explosion | SPEAKER.explosion | explosion |
| jump | SPEAKER.jump | jump |
| laser | SPEAKER.laser | laser |
| level-up | SPEAKER.level-up | level-up |
| low-energy | SPEAKER.low-energy | low-energy |
| prompt-tone | SPEAKER.prompt-tone | prompt-tone |
| right | SPEAKER.right | right |
| wrong | SPEAKER.wrong | wrong |
| ring | SPEAKER.ring | ring |
| score | SPEAKER.score | score |
| wake | SPEAKER.wake | wake |
| warning | SPEAKER.warning | warning |
| **03-Sonidos físicos** | metal-clash | SPEAKER.metal-clash | metal-clash |
| shot | SPEAKER.shot-1 | shot |
| glass-clink | SPEAKER.glass-clink | glass-clink |
| inflator | SPEAKER.inflator | inflator |
| running water | SPEAKER.running-water | running water |
| clockwork | SPEAKER.clockwork | clockwork |
| click | SPEAKER.click | click |
| current | SPEAKER.current | current |
| switch | SPEAKER.switch | switch |
| wood-hit | SPEAKER.wood-hit-3 | wood-hit |
| iron | SPEAKER.iron-1 | iron |
| drop | SPEAKER.drop | drop |
| bubble | SPEAKER.bubble-1 | bubble |
| wave | SPEAKER.wave | wave |
| magic | SPEAKER.magic | magic |
| spitfire | SPEAKER.spitfire | spitfire |
| heartbeat | SPEAKER.heartbeat | heartbeat |
| load | SPEAKER.load | load |

Simulación de instrumentos  
  
*cyberpi.audio.play\_music(note, beat, type = “piano”)*  
Reproduce la nota de los sonidos especificados en el tiempo especificado.

Esta API bloquea el hilo hasta que termine su reproducción.  
Puede usarse para compilar y reproducir música.  
  
Parámetros:

* Tipo *str:* tipo de instrumento a simular. El valor se setea en *piano.*
* nota *int:* frecuencia del sonido a reproducir. El rango de ajuste es de 0 a 132. La tabla “Valores y sus notas correspondientes” describe, como dice su nombre, los valores y sus notas correspondientes.
* beat *float* : duración de la nota que se va a tocar. Rango de ajuste: *beat>0.* En general, el tiempo de reproducción equivale a 1 segundo.

“Valores y sus notas correspondientes”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Value** | **Note** |  | **Value** | **Note** |  | **Value** | **Note** |
| 0 | C-1 | 43 | G2 | 86 | D6 |
| 1 | C#-1 | 44 | G#2 | 87 | D#6 |
| 2 | D-1 | 45 | A2 | 88 | E6 |
| 3 | D#-1 | 46 | A#2 | 89 | F6 |
| 4 | E-1 | 47 | B2 | 90 | F#6 |
| 5 | F-1 | 48 | C3 | 91 | G6 |
| 6 | F#-1 | 49 | C#3 | 92 | G#6 |
| 7 | G-1 | 50 | D3 | 93 | A6 |
| 8 | G#-1 | 51 | D#3 | 94 | A#6 |
| 9 | A-1 | 52 | E3 | 95 | B6 |
| 10 | A#-1 | 53 | F3 | 96 | C7 |
| 11 | B-1 | 54 | F#3 | 97 | C#7 |
| 12 | C0 | 55 | G3 | 98 | D7 |
| 13 | C#0 | 56 | G#3 | 99 | D#7 |
| 14 | D0 | 57 | A3 | 100 | E7 |
| 15 | D#0 | 58 | A#3 | 101 | F7 |
| 16 | E0 | 59 | B3 | 102 | F#7 |
| 17 | F0 | 60 | C4 | 103 | G7 |
| 18 | F#0 | 61 | C#4 | 104 | G#7 |
| 19 | G0 | 62 | D4 | 105 | A7 |
| 20 | G#0 | 63 | D#4 | 106 | A#7 |
| 21 | A0 | 64 | E4 | 107 | B7 |
| 22 | A#0 | 65 | F4 | 108 | C8 |
| 23 | B0 | 66 | F#4 | 109 | C#8 |
| 24 | C1 | 67 | G4 | 110 | D8 |
| 25 | C#1 | 68 | G#4 | 111 | D#8 |
| 26 | D1 | 69 | A4 | 112 | E8 |
| 27 | D#1 | 70 | A#4 | 113 | F8 |
| 28 | E1 | 71 | B4 | 114 | F#8 |
| 29 | F1 | 72 | C5 | 115 | G8 |
| 30 | F#1 | 73 | C#5 | 116 | G#8 |
| 31 | G1 | 74 | D5 | 117 | A8 |
| 32 | G#1 | 75 | D#5 | 118 | A#8 |
| 33 | A1 | 76 | E5 | 119 | B8 |
| 34 | A#1 | 77 | F5 | 120 | C9 |
| 35 | B1 | 78 | F#5 | 121 | C#9 |
| 36 | C2 | 79 | G5 | 122 | D9 |
| 37 | C#2 | 80 | G#5 | 123 | D#9 |
| 38 | D2 | 81 | A5 | 124 | E9 |
| 39 | D#2 | 82 | A#5 | 125 | F9 |
| 40 | E2 | 83 | B5 | 126 | F#9 |
| 41 | F2 | 84 | C6 | 127 | G9 |
| 42 | F#2 | 85 | C#6 |  |  |

Programa de ejemplo  
  
import cyberpi

import time

cyberpi.console.println(‘presione B para reproducir música’)

cyberpi.console.println(‘Presione B para reproducir música.’)

while True:

while not cyberpi.controller.is\_press(‘b’)

pass

*#Puede hacer clic con el boton derecho en un bloque para ver la información de ayuda.  
#El bloque usa codigos MIDI para indicar notas, en este ejemplo se usan los siguientes codigos MIDI:*   
  
*#Nota musical numerada/ Nota/Código MIDI  
# 1 C4 60*

*# 2 D4 62*

*# 3 E4 64*

*# 4 F4 65*

*# 5 G4 67*

*# 6 A4 69*

*# 7 B4 71*

cyberpi.audio.play\_music(60, 0.25)

cyberpi.audio.play\_music(60, 0.25)

cyberpi.audio.play\_music(67, 0.25)  
cyberpi.audio.play\_music(67, 0.25)  
cyberpi.audio.play\_music(69, 0.25)  
cyberpi.audio.play\_music(69, 0.25)  
cyberpi.audio.play\_music(67, 0.25)

time.sleep(1)

cyberpi.audio.play\_music(65, 0.25)

cyberpi.audio.play\_music(65, 0.25)

cyberpi.audio.play\_music(64, 0.25)

cyberpi.audio.play\_music(64, 0.25)

cyberpi.audio.play\_music(62, 0.25)

cyberpi.audio.play\_music(62, 0.25)

cyberpi.audio.play\_music(60, 0.25)

(simbolo)MP cyberpi.audio.play\_drum(type, beat)

Reproduce un sonido específico a una velocidad específica.  
  
tipo *str*: tipo de instrumento a ser simulado:

Rango de ajuste: }  
-snare: snare drum

-bass-drum: bass drum

-side-stick: beating the edge of a drum

-crash-cymbal: crash cymbal

-open-hi-hat: open hi-hat

-closed-hi-hat: closed hi-hat

-tambourine: tambourine

-hand-clap: handclap

-claves: claves

Beat *float*: Lo que durará la nota a tocar. este valor debe ser mayor que 0. Para una velocidad de reproducción general, un beat equivale a un segundo.

**Programa de ejemplo:**

import event, time, cyberpi  
  
@event.start():

def on\_start

global frequency

while True:

if cyberpi.is\_tiltforward():

cyberpi.audio.play\_drum(‘snare’, 0.25)

if cyberpi.is\_tiltforward():

cyberpi.audio.play\_drum(‘bass-drum’, 0.25)

if cyberpi.is\_tiltforward():

cyberpi.audio.play\_drum(‘open-hi-hat’, 0.25)

if cyberpi.is\_tiltforward():

cyberpi.audio.play\_drum(‘closed-hi-hat’, 0.25)

**Reproducción de sonidos grabados**

Comienza a grabar sonidos: Cuando la API esté siendo ejecutada, Cyberpi comenzará a grabar sonidos hasta que se ejecute audio.stop\_record(), o si la grabación dura más de 10 segundos. El archivo quedará guardado en la RAM, y esta se perderá si la cyberpi es apagada o reiniciada.

* cyberpi.audio.stop\_record()

*→*Detener la grabación de sonido:  
 Esta API es necesaria ser usada en combinación con audio.record()

* cyberpi.audio.play\_record\_until() *→*Reproduce el sonido grabado.  
  Esta API bloquea el hilo hasta que se termina de reproducir.

cyberpi.audio.play\_record()

*→*Reproduce el sonido grabado.  
  
**Programa de ejemplo:**

import time, event, cyberpi

@event.start

def on\_start():

“””Presione el centro del joystick para comenzar a grabar, y para detener la grabación se presiona el botón A. y para reproducir lo grabado, se presiona el botón B.   
  
Mueva el joystick de arriba a abajo para cambiar la velocidad de reproducción. mueva el joystick de un lado a otro para subir o bajar el volumen.”””

cyberpi.display.set\_brush(255, 255, 255)

cyberpi.led.set\_bri(30)

cyberpi.audio.set\_vol(50)

cyberpi.audio.set\_tempo(100)

cyberpi.console.println('1.Magical Recorder')

cyberpi.console.println('Presione en medio del joystick para iniciar la grabación')

cyberpi.console.println('')

cyberpi.led.show('orange orange orange orange orange')

@event.is\_press('middle')

def is\_press():

cyberpi.console.clear()

cyberpi.console.println('Presione el boton A para iniciar la grabación')

cyberpi.console.println('')

cyberpi.led.show('green green green green green')

cyberpi.audio.record()

@event.is\_press('a')

def on\_received():

cyberpi.console.clear()

cyberpi.console.println('presione el boton B para reproducir la grabación')

cyberpi.console.println('Mueva el joystick de arriba a abajo para cambiar ala velocidad de reproducción y muevelo de izquierda a derecha para cambiar el volumen de reproducción')

cyberpi.console.println('')

cyberpi.led.show('red red red red red')

cyberpi.audio.stop\_record()

@event.is\_press('b')

def on\_received1():

cyberpi.console.clear()

cyberpi.console.println('Playing...')

cyberpi.console.println('Mueva el joystick de arriba a abajo para cambiar ala velocidad de reproducción y muevelo de izquierda a derecha para cambiar el volumen de reproducción')

cyberpi.console.println('')

cyberpi.led.show('blue blue blue blue blue')

cyberpi.audio.play\_record\_until()

@event.is\_press('up')

def is\_press1():

"""Move the joystick up or down to change the playing speed.

The speed setting function is a feature of CyberPi's audio system, allowing you to have fun by slowing down or speeding up the playing.

The speed setting function can also be used to slow down or speed up the playing of the preset audio files and simulated instrument sounds."""

cyberpi.led.off(0)

cyberpi.audio.add\_tempo(10)

cyberpi.console.println(str('Playing speed:') + str(cyberpi.audio.get\_tempo()))

cyberpi.console.println('')

@event.is\_press('down')

def is\_press2():

cyberpi.led.off(0)

cyberpi.audio.add\_tempo(-10)

if cyberpi.audio.get\_tempo() < 40:

cyberpi.audio.set\_tempo(40)

cyberpi.console.println(str('Playing speed:') + str(cyberpi.audio.get\_tempo()))

cyberpi.console.println('')

@event.is\_press('left')

def is\_press3():

"""Move the joystick to the left or right to change the playing volume"""

cyberpi.led.off(0)

cyberpi.audio.add\_vol(-10)

cyberpi.console.println(str('Playing volume:') + str(cyberpi.audio.get\_vol()))

cyberpi.console.println('')

@event.is\_press('right')

def is\_press4():

cyberpi.led.off(0)

cyberpi.audio.add\_vol(10)

cyberpi.console.println(str('Playing volume:') + str(cyberpi.audio.get\_vol()))

cyberpi.console.println('')

**Simulador de Zumbido**

cyberpi.audio.play\_tone(freq, t)Simula el sonido de un zumbido

Parametros:  
  
freq *int* : Frecuencia del sonido a reproducir. rango de seteo: 20-500Hz. Por bien de nuestra salud auditiva, no reproducir sonidos fuera de ese rango. si desean cambiar la música cambiando la frecuencia, consulte las siguientes notas y su frecuencia correspondiente:

['C2','65'], ['D2','73'], ['E2','82'], ['F2','87'],

['G2','98'], ['A2','110'], ['B2','123'], ['C3','131'],

['D3','147'], ['E3','165'], ['F3','175'], ['G3','196'],

['A3','220'], ['B3','247'], ['C4','262'], ['D4','294'],

['E4','330'], ['F4','349'], ['G4','392'], ['A4','440'],

['B4','494'], ['C5','523'], ['D5','587'], ['E5','659'],

['F5','698'], ['G5','784'], ['A5','880'], ['B5','988'],

['C6','1047'], ['D6','1175'], ['E6','1319'], ['F6','1397'],

['G6','1568'], ['A6','1760'], ['B6','1976'], ['C7','2093'],

['D7','2349'], ['E7','2637'], ['F7','2794'], ['G7','3136'],

['A7','3520'], ['B7','3951'], ['C8','4186'], ['D8','4699'],

-t: duración del sonido a reproducir, rango de seteo : t>= 0   
  
Programa de ejemplo:  
  
*import event, time, cyberpi*@event.is\_press(‘a’)

def on\_received():  
 global frequency

for count in range(2):  
 cyberpi.audio.play\_tone(800,1)  
 cyberpi.audio.play\_tone(200,1)

**Velocidad y volumen del altavoz de la Cyberpi**

Puedes usar las siguientes APIs para para configurar la velocidad de reproducción y volumen del altavoz del cyberpi.  
  
*cyberpi.audio.add\_tempo(pct)*Cambia la velocidad de reproducción del altavoz de la cyberpi   
  
Parámetro:  
  
-pct *int* : porcentaje (de la velocidad de reproducción normal) por el cual se va a cambiar la velocidad de reproducción , si es negativo es porque habrá una disminución de velocidad, y si es positivo, es porque se acelerará.   
  
cyberpi.audio.set\_tempo(pct)  
*→*establece la velocidad de reproducción del altavoz   
  
Parámetro:  
-pct *int*: Porcentaje de velocidad normal, con rango de ajuste en 25-400%

cyberpi.audio.get\_tempo()*:* Te dice la velocidad a la cual está el altavoz en ese momento. devolviendo un número entero entre 25 a 400%

cybepi.audio.add\_vol(val) *→*Cambia el volumen del altavoz de la Cyberpi  
  
Parámetro:  
-pct:*int* porcentaje por el cual se cambia el volumen, va de un rango de -100 a +100, Valor negativo es que se baja el volumen, y positivo que se sube.

cyberpi.audio.set\_vol(val)

→establece el nivel de volumen de la Cyberpi  
  
parámetro:   
 pct: *int* , porcentaje del volumen de reproducción , en un rango de 0 a 100%  
  
cyberpi.audio.get\_vol() *→*obtiene el volumen de la cyberpi, el valor devuelto es un entero entre 0 y 100%.  
  
cyberpi.audio.stop() *→*Detiene todoslos sonidos  
  
**Luces LEDS**

Cyberpi está equipado con 5 leds programables, puedes usar las siguientes APIs para compilar un código que controle los leds.  
  
cyberpi.led.on(r, g, b, id = “all”) *→*Establece los colores de los leds.  
  
Parámetros:

-r: puede ser *int o str*

-r: *int* intensidad del rojo, va en un rango de 0 a 255  
-r: *str* nombre completo o abreviatura de algún color, a continuación se describirán los colores y su abreviaturas:

1.-red (rojo) r  
2.-orange(naranjo) o  
3.-Yellow(amarillo) y

4.-green(verde) g  
5.-cyan c  
6.-blue(azul) b  
7.-purple(morado) p

8.-white (blanco) w  
9.-black (negro) k

-g: *int* intensidad del verde, rango de 0 a 255.  
-id *int* o *str* el valor por defecto es *all*

-id *str* solo el valor *all* es valido

Posición de los LEDs



Ejemplo de programa:

import cyberpi  
cyberpi.led.on(255,0,0) #Enciende todos los LEDs

import cyberpi

cyberpi.led.on(red, id=1) #enciende el LED 1 en rojo  
  
import cyberpo

cyberpi.led.on(‘r’, id = 1) #enciende el LED 1 en rojo

cyberpi.led.play(name = “arcoiris”

*→*Muestra la animación de LED especificada  
esta API bloquea el hilo hasta que termine.

parametro:

Name: *str* , nombre de la animaci+on LED, el nombre de predeterminado es *arcoiris*, y existen varios más:

1.-rainbow

2.-spoondrift

3.-meteor\_blue

4.-meteor\_green

5.-flash\_red

6.-flash\_orange

7.-firefly

cyberpi.led.show(color) *→*establecer el o los colores de los 5 leds  
  
Parámetros:   
  
-color: *srt* colores de los 5 leds, numerados como *color1, color2, color3, color4, color5,* con un espacio entre dos colores cualquiera. Si se establecen más de 5 colores, solo se guardarán los primeros 5. se puede establecer este nombre en el nombre completo del color, o en su abreviatura.

led.move(step=1)

*→*Hace que los colores de los LEDs se desplacen de izquierda a derecha el número de veces especificado.

Parámetro:  
  
-step *int* número de posiciones por las que pasan los colores de los LEDs, el rango de ajuste va de -4 a +4, con un valor predeterminado de 1.  
Por ejemplo, configura los cinco LEDs en r, o, y, g, c y usa la API. Cuando se establece en el paso 0, los colores de los led se verán como la figura 1, luego en el paso 1 se verán como la figura 2.



cyberpi.led.off(id = “all”)

*→*Apaga los LEDs especificados.   
  
Parámetro:

-id *int* o *str* posición del LED a apagar, su valor por defecto es *all*, indicando a todos los LEDs a que se apaguen, para elegir un LED, debe colocar un número de 1 a 5, correspondiendo a los 5 LEDs.  
  
cyberpi.led.add\_bri(Brightness) Cambia el brillo de los LEDs de la cyberpi   
  
parámetro:  
  
brigtness *int* porcentaje en el que se va a cambiar el brillo, con un rango de configuración de -100 a +100, número negativo es bajar el brillo, y positivo es subirlo.

*cyberpi.led.set\_bri(brightness)*

Establece el brillo de los LEDs de la Cyberpi.  
parámetro:  
brightness *int* brillo de los LEDs , el porcentaje de configuración va de 0 a 100%

*cyberpi.led.get\_bri()*

obtiene el brillo de los LEDs de la cyberpi

devuelve el valor del brillo en un rango de 0 a 100.

Programa ejemplo:

*import cyberpi*

*while True:  
 for count in range(10):  
 cyberpi.led.add\_bri(10)  
 cyber.led.move(1)*

*for count2 in range(10):  
 cyberpi.led.add\_bri(-10)  
 cyberpi.led.move(1)*

Display

La cyberpi está equipada con una pantalla a todo color de 1,89 pulgadas. con las siguientes APIs, se puede usar la cyberpi para mostrar textos e imágenes, dibujar graficos, diseñar juegos y crear aplicaciones.   
  
Texto y aplicaciones  
  
*cyberpi.console.print(mensaje)*

Muestra textos en la pantalla de la cyberpi, con saltos de línea automáticos.

Cuando se ejecuta este bloque, los textos se muestran en la misma línea, y cuando una línea está llena, el texto continúa en la siguiente línea.

parámetro:  
  
mensaje *str* es el texto que será mostrado, un ejemplo sería “Hola”  
  
Programa de ejemplo  
  
*import cyberpi  
cyberpi.console.print(‘Hello’)  
cyberpi.console.print(‘World’)*

*cyberpi.console.println(mensaje)*

Muestra textos en la pantalla de la cyberpi con saltos de línea forzados.

mensaje *str* es el texto que será mostrado, un ejemplo sería “Hola”.

*import cyberpi*

*cyberpi.console.println(‘Hello world’)*

*cyberpi.console.println(‘Hello’ +’ ‘+’world’)*

Posición de los textos

*cyberpi.display.show\_label(mensaje, tamaño, x, y )*

muestra un texto en una posición específica en la pantalla de la cyberpi.

Parámetro:

mensaje: *str,* texto a ser mostrado, como *hola*

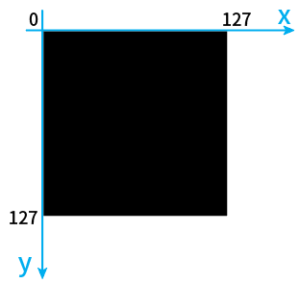
tamaño: *int*, tamaño de fuente del texto a mostrar, como *12, 24, 36,* etc.

x: *str o int,* es la posición donde se mostrará el texto.  
  
Cuando x es *str,* las opciones son las siguientes:

*top\_mid:* centro superior *top\_med:* superior izquierda *top\_right:* superior derecha *center:* en el centro *mid\_left:* al medio a la izquierda *mid\_right:*al medio a la derecha

*bottom\_mid:* en el centro inferior *bottom\_left:* Inferior a la izquierda *botoom\_right:* inferior a la derecha

Cuando x es *int,* el valor va de 0 a 128, indicando la coordenada x de la esquina superior izquierda del texto.

-y: *int*  la coordenada y va de la esquina superior izquierda de la pantalla, en la siguiente imagen se muestran como van las coordenada x e y en la pantalla 

Programa de ejemplo

*import cyberpi*

*cyberpi.display.show\_label(‘cyberpi’, 16, ‘center’)*

*inport cyberpi  
cyberpi.display.show\_label(‘cyberpi’, 16, x= 64, y= 64)*

Charts  *cyberpi.linechat.add(data)*Añade un dato y crea un gráfico de línea.  
  
Pueden usar la API *cyberpi.display.set\_brush(color)* para poder establecer el color de una línea y , por lo tanto, puede dibujar gráficos de líneas de diferentes colores, para diferenciar distintos conjuntos de datos.

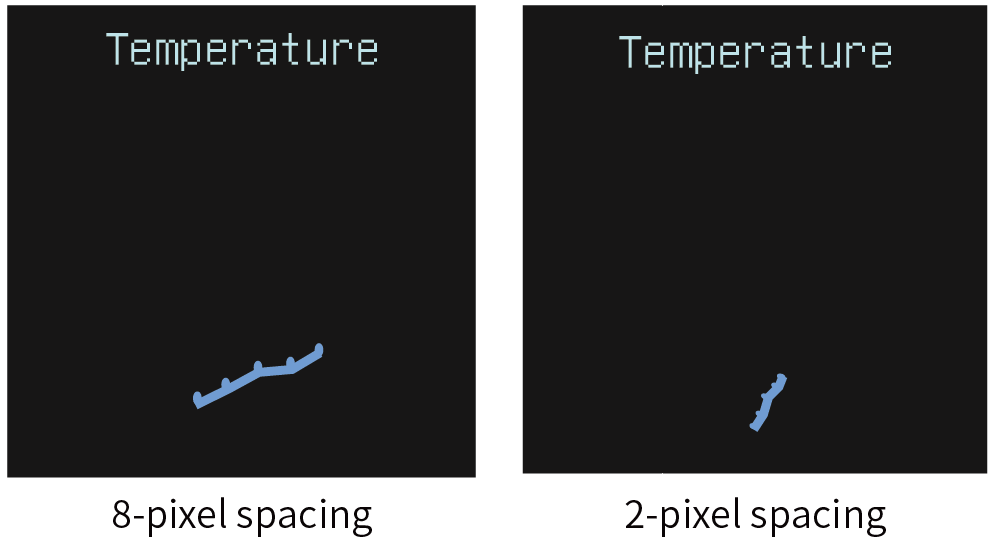
parámetro:   
- data: *float* rango de 0 a 100.

si un valor que quieren ingresar se sale del rango, deben volver a escalar todos los datos para que así puedan estar dentro del rango aceptado.

por ejemplo, puedes reescalar 200, 300 y 400 a 20, 30 y 40, respectivamente.

*cyberpi.linechat.set\_step(step)*

establece el espacio entre datos del gráfico de líneas

-step: *int*, espaciado entre puntos de datos, el rango de configuración va de 0 a 128 (son los pixeles).  
por ejemplo en la siguientes imágenes se pueden observar 2 graficos de lineas con los mismos datos, pero con diferente espaciado entre datos. 

*cyberpi.barchat.add(data)*

Agrega un dato y crea un gráfico de barras.  
  
Pueden usar la api *display.set\_brush(r, g , b)* para configurar el color de cada barra, y así poder crear un gráfico con cada grupo de datos diferenciado cada uno por un color distinto. El ancho de las barras cambia con el número de barras a mostrar.

Parámetro:  
- data: *float,* rango de 0 a 100.

Al igual que con el gráfico de líneas, si el rango del dato está fuera de la escala, se deben reescalar todos los datos para que estén en el rango.  
  
Por ejemplo, puedes reescalar 200, 300 y 400 a 20, 30 y 40, respectivamente.

*cyberpi.piechart.add(data)*

Añade un dato y crea un gráfico de torta, o circular.

Pueden usar la api *display.set\_brush(r, g , b)* para configurar el color de cada sección, y así poder crear un gráfico con cada grupo de datos diferenciado cada uno por un color distinto. El tamaño de las secciones cambia con el número de piezas de datos.

Parámetro:  
-data: *float,* va de 0 a infinito.

*cyberpi.table.add(fila, columna, data)*

añade datos y muestra una tabla.

El número de filas y columnas depende de la cantidad de datos que se ingrese.  
Se admite un máximo de 4 filas x 3 columnas. cuando un dato es demasiado largo, se desplaza a la siguiente celda.  
  
Puedes usar la api *display.set\_brush(r, g, b)* para configurar el color de cada trozo, el contenido pueden ser textos, o iconos predefinidos.  
  
Parámetros:  
-row *int* número de filas a crear.  
-column *int* , número de columnas a crear.

-data: *str* contenido a introducir en la tabla

Programa de ejemplo  
  
*import random, cyberpi, event, time*

*@event.start*

*def on\_start():  
 cyberpi.console.println(‘2.Volumn-reactive Lights and Columns’)*

*cyberpi.console.println(‘2.光影音量柱’)*

*cyberpi.console.println(‘Press B to Start’)*

*cyberpi.console.println(‘按 B 以开始...’)*

*while not cyberpi.controller.is\_press(‘b’)  
 cyberpi.display.set\_brush(random.randint(1, 255), random.randint(1, 255), random.randint(1, 255))  
  
 cyberpi.broadcast('Message 1')*

*@event.receive('Message 1')*

*def on\_receive():*

*"""The bar chart changes with the volume of the sounds in the ambient environment. Try playing a song!*

*The brightness of the LEDs also changes with the volume."""*

*cyberpi.led.on(114, 0, 255, 1)*

*cyberpi.led.on(191, 0, 255, 2)*

*cyberpi.led.on(255, 0, 195, 3)*

*cyberpi.led.on(255, 0, 72, 4)*

*cyberpi.led.on(255, 0, 0, 5)*

*while True:*

*cyberpi.led.set\_bri(cyberpi.get\_loudness("maximum"))*

*cyberpi.display.set\_brush(114, 0, 255)*

*cyberpi.barchart.add(cyberpi.get\_loudness("maximum"))*

*time.sleep(0.02)*

*cyberpi.display.set\_brush(191, 0, 255)*

*cyberpi.barchart.add(cyberpi.get\_loudness("maximum"))*

*time.sleep(0.02)*

*cyberpi.display.set\_brush(255, 0, 195)*

*cyberpi.barchart.add(cyberpi.get\_loudness("maximum"))*

*time.sleep(0.02)*

*cyberpi.display.set\_brush(255, 0, 72)*

*cyberpi.barchart.add(cyberpi.get\_loudness("maximum"))*

*time.sleep(0.02)*

*cyberpi.display.set\_brush(255, 24, 24)*

*cyberpi.barchart.add(cyberpi.get\_loudness("maximum"))*

*time.sleep(0.05)*

*cyberpi.display.set\_brush(r, g, b)*-r *int o str*r: *int,* intensidad del color ro, en el rango de 0 a 255  
r: *str,* nombre completo o abreviatura del color, a continuación se muestran los colores y sus abreviaturas:

*red r*

*orange o*

*yellow y*

*green g*

*cyan c*

*blue b*

*purple p*

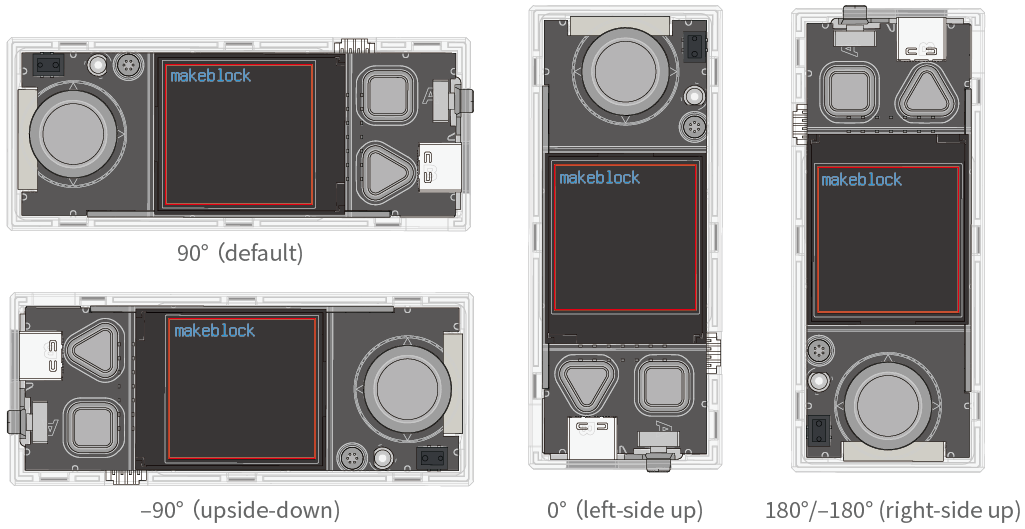
*white w*

*black k*

.g: *int,* intensidad del color verde, va de 0 a 255, y así con el resto de los colores.

Display de direcciones  
  
*cyber.display.rotate\_to(angle)*Establece la dirección de visualización de la pantalla de la cyberpi (angulo)  
  
Parámetro:  
-angle: *int,* dirección de visualización*,* rango de ajuste: -90°, 0°, 90°, 180° y -180°.

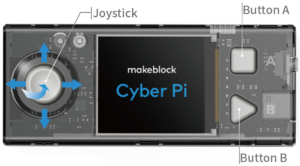
las direcciones de visualización correspondientes se observan en las siguientes imagenes

**

*cyberpi.display.clear()*Limpia la pantalla de la cyberpi

*cyberpi.display.off()*apaga la luz de fondo de la pantalla de la cyberpi

Sensores  
  
Con las siguientes APIs, pueden obtener valores de salida de los sensores en su cyberpi o información en su cyberpi.



Las siguientes APIs están disponibles para poder programar la cyberpi.

*cyberpi.controller.is\_press()*

Determina si se presiona el botón A, el B o la dirección en la que el joystick se está moviendo.

parámetro:  
  
-name: *str,*  nombre del botón o la dirección del joystick.  
  
Rango de ajuste:

*a*: botón A (en el botón cuadrado)

*b*: botón B (en el botón triangular)

*up*: joystick se mueve hacia arriba

*down*: joystick se mueve hacia abajo

*left*: joystick se mueve a la izquierda

*right*: joystick se mueve a la derecha

*middle*: joystick se presiona en el centro

any\_direction: joystick se mueve en cualquier dirección

any\_button: botón A o B

any: cualquier boton o direciión del joystick

se devuelve un valor booleano

*cyberpi.controller.get\_count(name)*

Obtiene el número de veces en que un botón es presionado o el joystick es movido a una dirección en particular.

Parámetro:

-name: *str,* nombre del botón o la dirección del joystick.

*a*: botón A (en el botón cuadrado)

*b*: botón B (en el botón triangular)

*up*: joystick se mueve hacia arriba

*down*: joystick se mueve hacia abajo

*left*: joystick se mueve a la izquierda

*right*: joystick se mueve a la derecha

*middle*: joystick se presiona en el centro

any\_direction: joystick se mueve en cualquier dirección

any\_button: botón A o B

any: cualquier boton o direciión del joystick

un valor entero es devuelto

cyberpi.controller.reset\_count(name)  
  
resetea el número de veces en que un botón es presionado o el joystick es movido a una dirección en particular.

Salidas de sensores  
  
*cyberpi.set\_bri()*

Obtienes el brillo de la luz ambiente detectada por el sensor de luz en cyberpi.

Se devuelve un valor entero *int*  que va de 0 a 100, cuando el valor devuelto es 100, se alcanza el límite superior de medición.

*cyberpi.get\_loudness(mode = “maximum”)*Obtiene volumen del sonido ambiente detectado por el micrófono en cyberpi.

Parámetro:  
-mode: *str,* tipo de sonoridad que desea obtener.

Rango de ajuste

*average:* sonoridad promedio en un periodo de tiempo.

*maximum:* sonoridad máxima en un período; obviamente este modo es preferible para proyectos sensibles al sonido.

Se devuelve un valor entero *int*  que va de 0 a 100, cuando el valor devuelto es 100, se alcanza el límite superior de medición del microfono.

Temporizador  
  
*cyberpi.timer.get()*

obtienes el valor del conteo del temporizador.

El contador empieza a contar cuando la cyberpi se enciende.

un valor flotante *float* es devuelto, en segundos.

*cyberpi.timer.reset()*

Resetea el valor del temporizador devolviéndole a 0.

Otras salidas  
 *cyberpi.get\_mac\_adress()*

Obtiene la dirección de control de acceso a medios (dirección MAC) del Wi-Fi.

Tengan en cuenta que la dirección obtenida puede ser incorrecta si la red Wi-Fi no está conectada.

El valor devuelto es un valor de cadena de 12 bytes, por ejemplo, FFEE33445566.

*cyberpi.get\_battery()*

Obtienes el nivel de batería (en %) en la pocket shield conectada a la cyberpi.

Se devuelve un valor *int* que va entre 0 y 100. Cuando se devuelve el valor 100, es porque la batería está en su nivel máximo; y cuando se devuelve 0, ningún pocket shield está conectado a CyberPi o la batería de la pocket shield conectada se quedó sin batería.

*cyberpi.get\_firmware\_version()*Obtienes el número de versión del firmware

El valor es devuelto como un *str*, por ejemplo 44.01.001

*cyberpi.get.ble()*Obtiene el nombre del dispositivo bluetooth utilizado por la cyberpi.

el valor devuelto es un *str*, por ejemplo, *makeblock\_LE\_XXXXX.*

*cyberpi.get\_name()*Obtienes el nombre de tu cyberpi  
  
El nombre que establezca se muestra en la interfaz de usuario del sistema CyberOS que se ejecuta en la cyberpi. Al configurar el nombre del dispositivo, los usuarios pueden encontrar su cyberpi o identificarse entre sí en una LAN o IoT.

Parámetro:  
  
-name: *str,* nombre de su cyberpi : alex, Tom.

*cyberpi.get\_language()*Obtienes el lenguaje que está siendo usado en la cyberpi.

El valor retornado es un *str*

*"chinese"*: Chino

*"cantonese"*: Cantonés

*"japanese"*: Japonés

*"english"*: Inglés

*"french"*: Francés

*"german":* Alemán

*"spanish":* Español

*"portuguese"*: Portugués

*"russian"*: Ruso

*"korean"*: Koreano

*"italian"*: Italiano

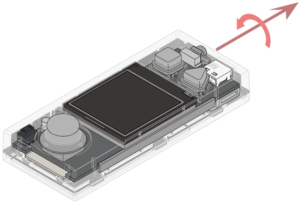
*"dutch"*: Holandés.

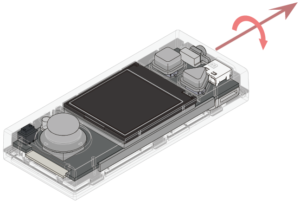
Detección de movimiento

Con el giroscopio, el acelerómetro y el algoritmo de detección de movimiento incorporado en la cyberpi, se puede implementar un control basado en el movimiento. Se pueden utilizar las siguientes APIs para obtener las posturas de la cyberpi.

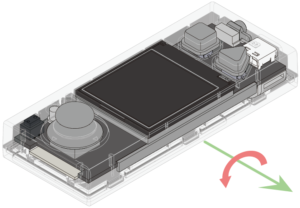
Posturas de la cyberpi

*cyberpi.is\_tiltforward()*Determina si la cyberpi está inclinada hacia adelante. La siguiente imagen muestra la inclinación hacia adelante:

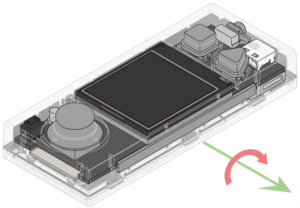
  
Un booleano *bool* es devuelto.  
  
*cyberpi.is\_titlback*Determina si la cyberpi está inclinada hacia atrás.



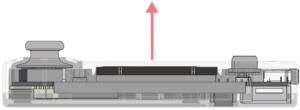
*cyberpi.is\_tiltleft()*  
Determina si la cyberpi está inclinada a la izquierda



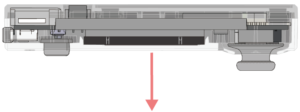
cyberpi.is\_tiltright()  
  
Determina si la cyberpi está inclinada a la derecha



*cyberpi.is\_faceup()*

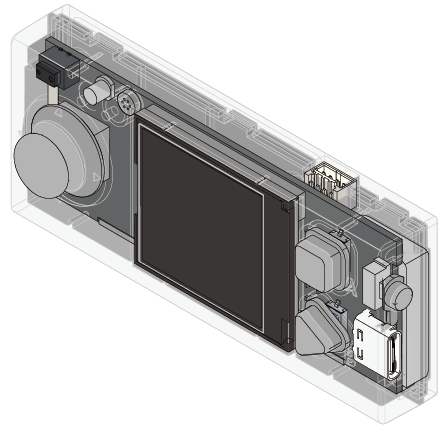
Determina si la cyberpi está boca arriba  
  


cyberpi.is\_facedown()  
  
Determina si la cyberpi está boca abajo.



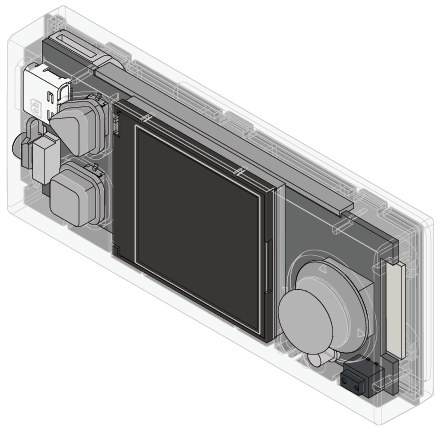
*cyberpi.is\_stand()*

Determina si la cyberpi se coloca perpendicular al suelo, con la tira de LED en la parte inferior, como se muestra en la siguiente figura:

**

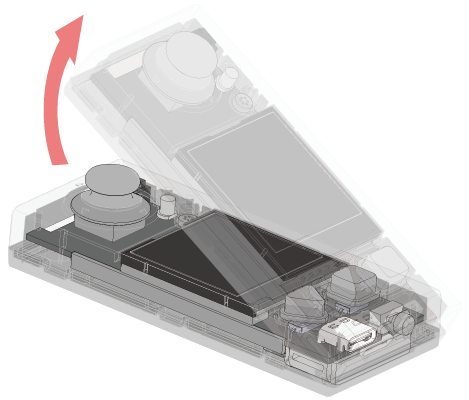
*cyberpi.is\_handstand()*

Determina si la cyberpi se coloca perpendicular al suelo, con la tira de LED en la parte superior, como se muestra en la siguiente figura:

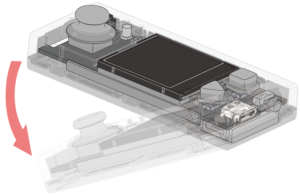
**

Movimientos continuos  
  
*cyberpi.is\_shake()*Se determina que la cyberpi se agita cuando la fuerza de agitación es de más de 20.

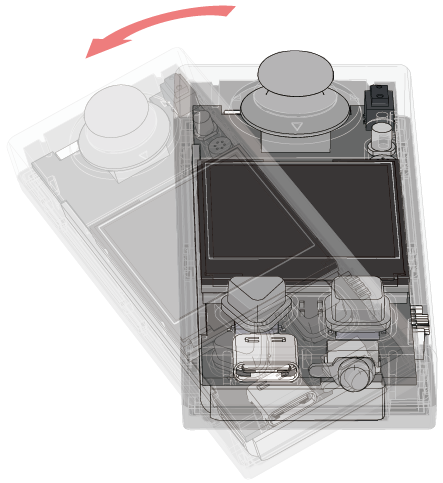
un valor booleano *bool* es devuelto.

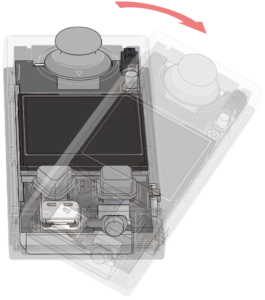


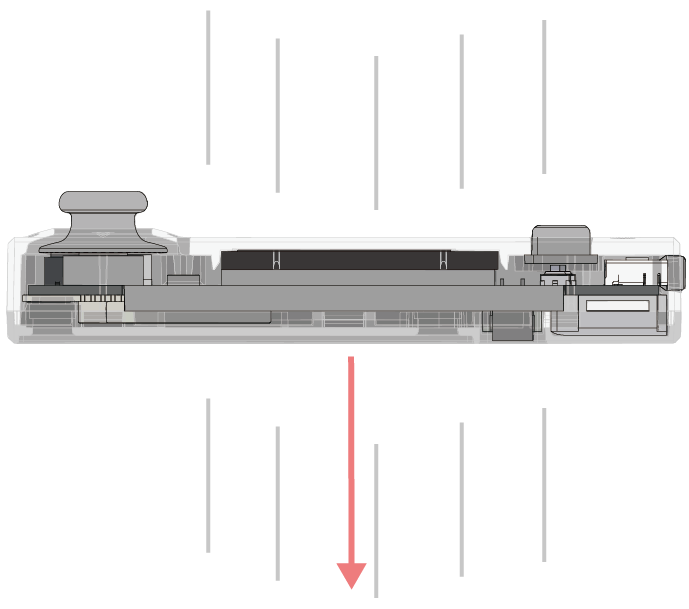
*cyberpi.is\_wavedown()*

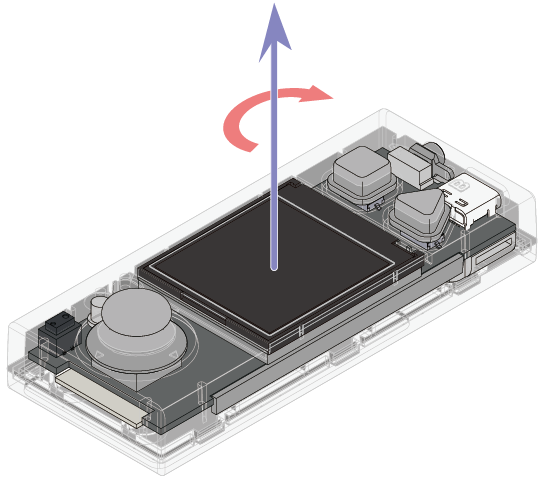
Determina si la cyberpi se agita hacia abajo.  
  


*Cyberpi.is\_waveleft()*Determina si la cyberpi se agita hacia la izquierda

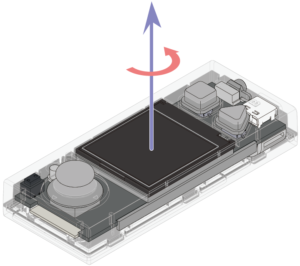
**

*cyberpi.is\_waveright()*determina si la cyberpi se agita hacia la derecha. **

*cyberpi.is\_freefall()*Determina cuando la cyberpi va cayendo.  
  
  
  
*cyberpi.is\_clockwise()*

Determina si la cyberpi gira en el sentido de las agujas del reloj alrededor del eje z.  


*cyberpi.is\_anticlockwise()*  
Determina si la cyberpi gira en el sentido contrario de las agujas del reloj alrededor del eje z.



Datos de postura y aceleración.

*cyberpi.get\_shakeval()*

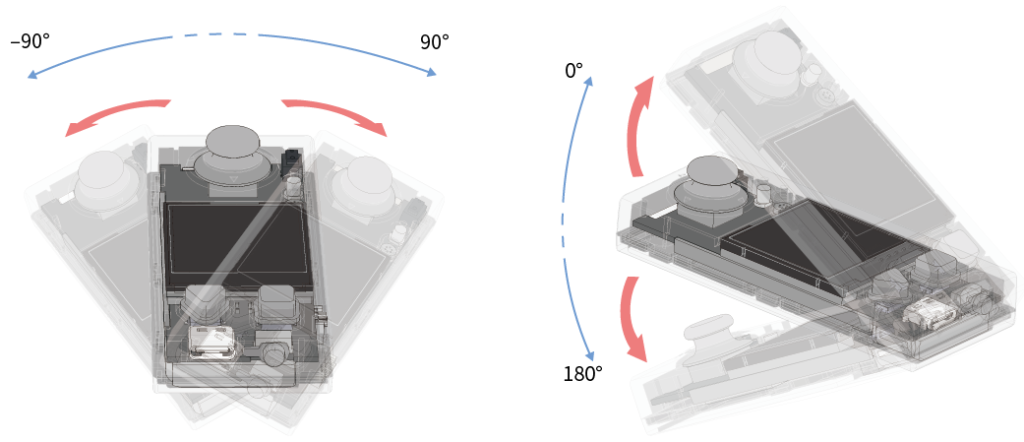
Obtienes la fuerza con la que está siendo agitado.

La fuerza de agitación está positivamente relacionada con la frecuencia y amplitud de agitación.

Se devuelve un valor *int* que oscila entre 0 y 100. Cuando el valor devuelto es 100, se alcanza el límite superior de fuerza de agitación.

*cyberpi.get\_wave\_angle()*Obtiene la dirección en la que se agita cyberpi

Se devuelve un valor *int* que oscila entre –179 y 180, en grados.



*cyberpi.get\_wave\_speed()*obtiene la velocidad a la que agita la cyberpi.

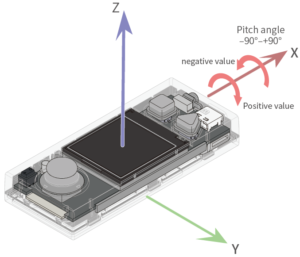
La velocidad está positivamente relacionada, pero no es igual a la velocidad de movimiento real de la cyberpi. Se puede causar un gran error en el valor obtenido debido al error integral.

Es devuelto un dato *int* en un rango de 0 a 100.  
 *cyberpi.get\_pitch()*

Obtiene el ángulo de inclinación de la cyber

El ángulo de inclinación se refiere al ángulo entre el eje y, y el plano horizontal.

Se devuelve un valor int que oscila entre –90 y 90, en grados.

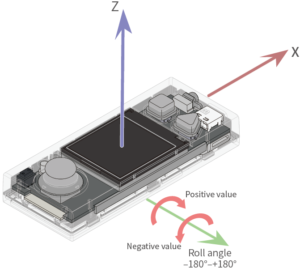


*cyberpi.get\_roll()*

Obtiene el ángulo de balanceo de la cyberpi.

El ángulo de balanceo se refiere al ángulo entre el eje x, y el plano horizontal.

Se devuelve un valor *int* que oscila entre –179 y 180, en grados.

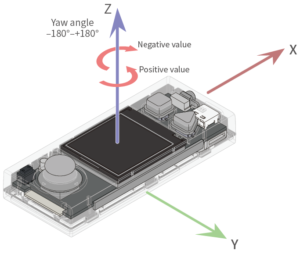


*cybepi.get\_yaw()*

Obtiene el ángulo de guiñada de CyberPi

El ángulo de guiñada se refiere al ángulo que CyberPi gira alrededor del eje z.

Se devuelve un valor int que oscila entre –180 y 180, en grados.

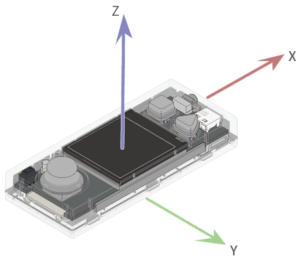


Nota: Pueden producirse errores acumulados en el ángulo de guiñada obtenido debido a la falta de brújula. Puede usar la API reset\_yaw() para calibrar el ángulo de guiñada. No hay brújula configurada en CyberPi y, por lo tanto, reset\_yaw() se usa para restablecer el ángulo de guiñada a 0.

*cyberpi.reset\_yaw()*

Restablece el ángulo de guiñada

No hay brújula configurada en CyberPi y, por lo tanto, reset\_yaw() se usa para restablecer el ángulo de guiñada a 0.

*cyberpi.get\_acc(axis)*

Obtiene la salida (en m/s²) del acelerómetro en el eje especificado

Parámetro:

eje: str, eje definido para la cyberpi; rango de ajuste: x, y, z

Se devuelve un valor flotante *float*. Deben tener en cuenta que el acelerómetro mide la gravedad de la Tierra. Cuando la cyberpi se coloca sobre un escritorio, la salida del acelerómetro en el eje z es –9,8 m/s² (1 g = 9,8 m/s²)

*cyberpi.get\_gyro(axis)*

Obtiene la velocidad angular alrededor del eje especificado.

Parámetro:  
-eje *str:* eje definido para la cyberpi;con un rango de ajuste: x, y, z

Se devuelve un valor int que oscila entre –500 y 500, en grados/segundos (°/s).

*cyberpi.get\_rotation(axis)*

Obtenemos el ángulo en el que gira la cyberpi alrededor de un eje especificado, con la dirección de giro a contrarreloj como el movimiento positivo.  
  
Parámetro:

-eje: *str*, es el eje definido para la cyberpi; rango de ajuste: x, y, z

Se devuelve un número, en grados.

*cyberpi.reset\_ rotation(axis = “all”)*

Restablece el o los ángulos que la cyberpi ha girado, a 0.

Después de ejecutar esta API, el valor obtenido por *get\_rotation(axis)* se cuenta desde cero.

Parámetro:

-axis: *str*, Uno o todos los ejes definidos para la cyberpii; rango de ajuste: *x, y, z*, y *all.*Wi fi  
  
La cyberpi viene equipada con un módulo wifi. Se pueden usar las siguientes APIs para conectar la cyberpi a Internet o realizar transmisiones LAN.  
  
*cyberpi.connet.wifi(ssid, password)*

Comienza la conexión wifi.  
  
Esta API no bloquea el hilo , por lo que la cyberpi no está necesariamente conectada a Internet cuando esta API se detiene. Debe usar *wifi.is\_connect()* para determinar si la conexión Wi-Fi está completa.  
  
Parámetro:

-ssid: *str*, cuenta Wi-Fi; configúralo en una cuenta que se pueda usar.

-password: *str*, contraseña de Wi-Fi; configúralo con la contraseña de la cuenta que usas.

*cyberpi.wifi.is\_connect()*Detecta el estado de la conexión Wi-Fi

Se devuelve un valor booleano *bool*.

*cyberpi.wifi\_broadcast.set(message, val)*

Envía una transmisión LAN

Para obtener más información, consulte Difusión de LAN.  
  
*cyberpi.wifi\_broadcast.get(mesage)*

Recibe una transmisión LAN

Para obtener más información, consulte Difusión de LAN.

Servicios de la nube.

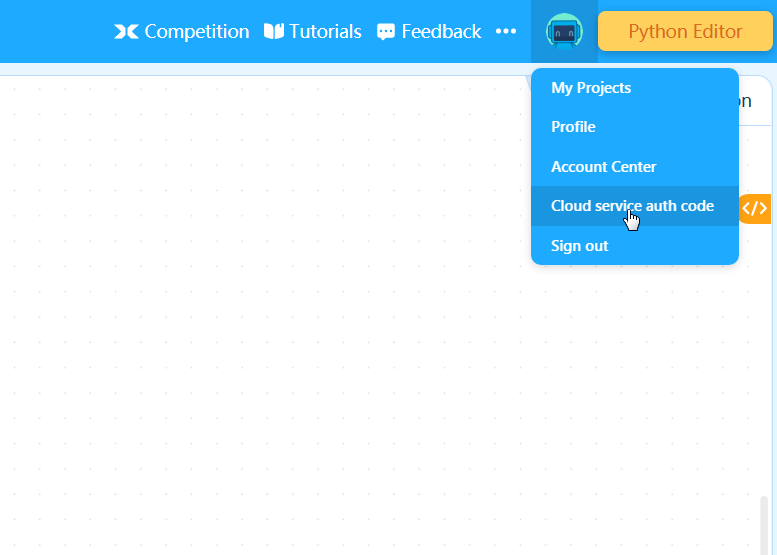
Después de conectar la cyberpi a Internet, pueden usar varios servicios en la nube proporcionados por Makeblock para implementar las funciones de IA e IoT de CyberPi.

*cyberpi.cloud.setkey(key)*

Use la API *cyberpi.cloud.setkey(clave)* para verificar su cuenta de mBlock, de modo que pueda utilizar los servicios en la nube proporcionados por Makeblock. Deben tener en cuenta que pueden usar esta API solo después de haber conectado CyberPi a Internet.

Parámetro:

-clave: *str*, código de autorización para servicios en la nube.

Inicie sesión en mBlock 5 en la web o en su cliente de PC y luego obtenga su código de autorización de servicio en la nube de la siguiente manera.  


IoT

*cyberpi.cloud.weather(option, woe\_id)*Obtiene los datos meteorológicos en tiempo real de la ubicación especificada

Parámetros:

-option: *str*, opción de datos meteorológicos

Rango de ajuste:

-*max\_temp*: temperatura más alta

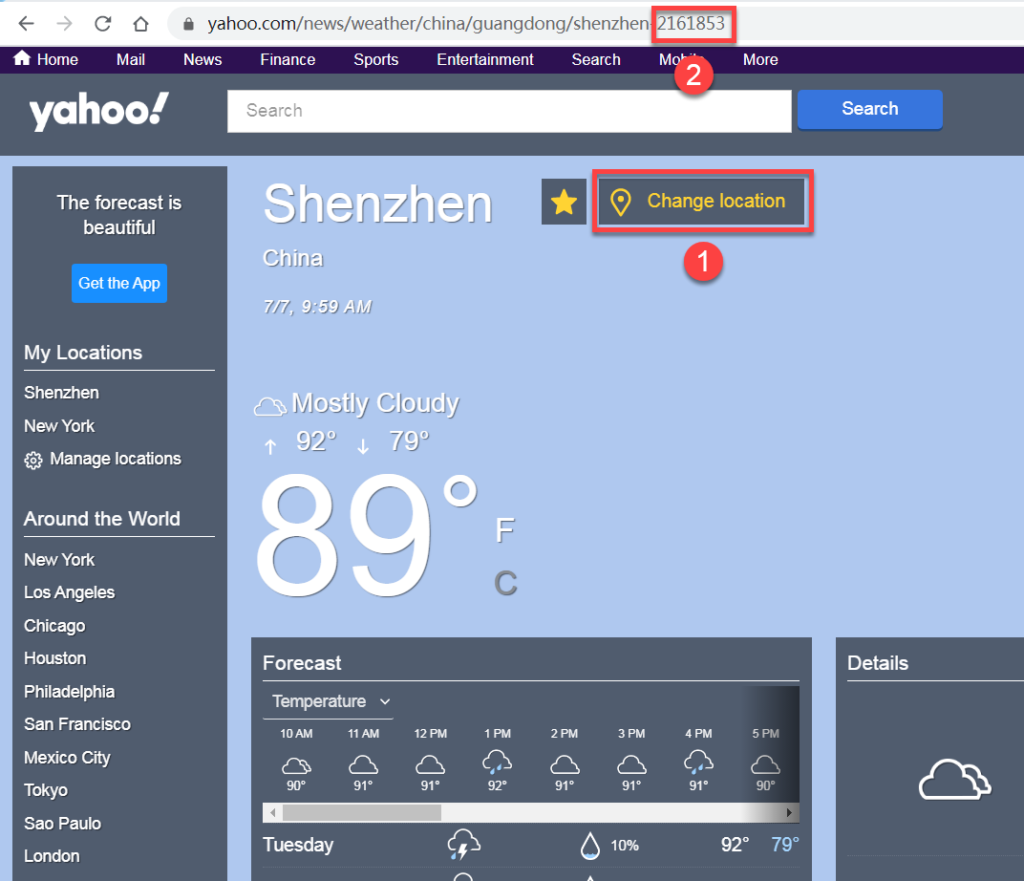
-*min\_temp*: temperatura más baja

-*clima*: temperatura

-*humedad*: humedad

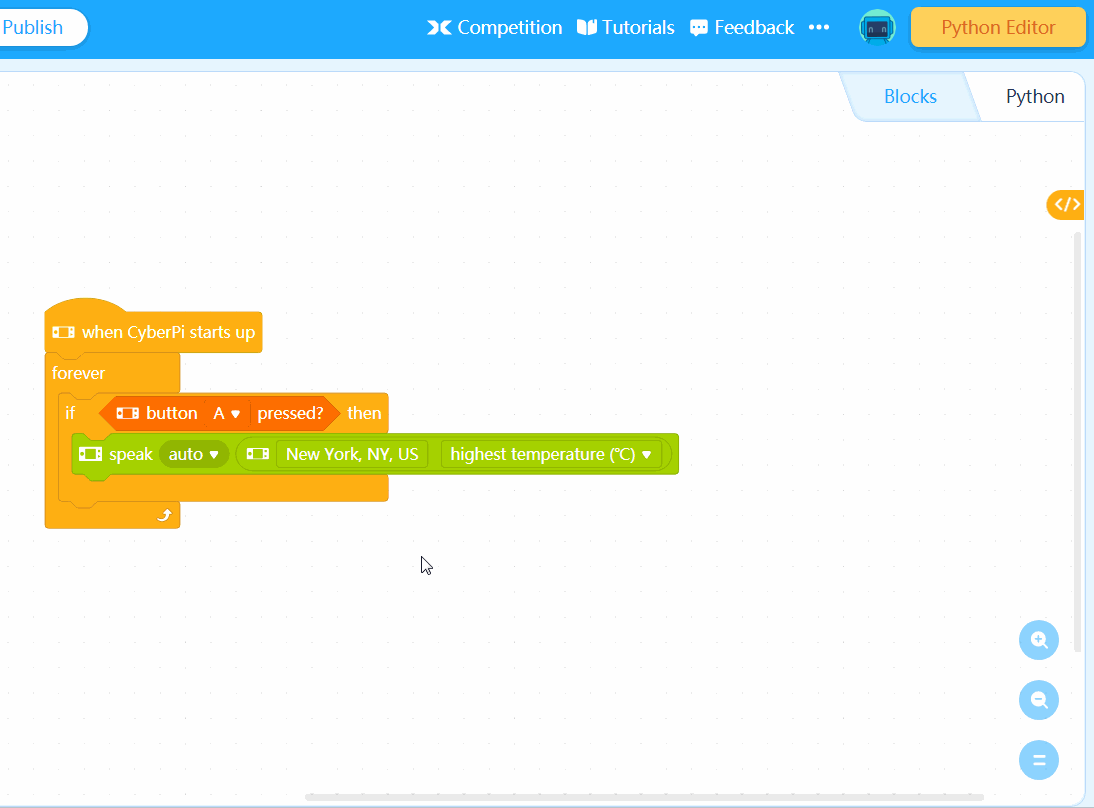
-*woe\_id*: *str,* código de la ciudad de la que se van a obtener los datos meteorológicos

Puedes visitar Yahoo Weather para obtener los códigos de las ciudades.



Se puede cambiar la ubicación a aquella en la que se buscará la información meteorológica y luego vea el código en la barra de direcciones.

Alternativamente, puede ver el código de una ciudad en el bloque correspondiente en mBlock 5, como se muestra en la siguiente figura.



Se devuelve un valor str, que proporciona la información meteorológica.

El valor devuelto puede ser uno de los siguientes:

0 tornado

1 tropical storm

2 hurricane

3 severe thunderstorms

4 thunderstorms

5 mixed rain and snow

6 mixed rain and sleet

7 mixed snow and sleet

8 freezing drizzle

9 drizzle

10 freezing rain

11 showers

12 rain

13 snow flurries

14 light snow showers

15 blowing snow

16 snow

17 hail

18 sleet

19 dust

20 foggy

21 haze

22 smoky

23 blustery

24 windy

25 cold

26 cloudy

27 mostly cloudy (night)

28 mostly cloudy (day)

29 partly cloudy (night)

30 partly cloudy (day)

31 clear (night)

32 sunny

33 fair (night)

34 fair (day)

35 mixed rain and hail

36 hot

37 isolated thunderstorms

38 scattered thunderstorms

39 scattered showers (day)

40 heavy rain

41 scattered snow showers (day)

42 heavy snow

43 blizzard

44 not available

45 scattered showers (night)

46 scattered snow showers (night)

47 scattered thundershowers

*cyberpi.cloud.air(option, woe\_id)  
  
-*option: *str*, opción de calidad del aire

Rango de ajuste:

*-aqi*: índice de calidad del aire

*-pm2.5*: densidad de PM2.5

*-pm10*: densidad de PM10

*-co*: densidad de CO

*-so2*: densidad de SO2

*-no2*: densidad de NO2

*-woe\_id: str*, código de la ciudad de la que se van a obtener los datos meteorológicos

Se devuelve un valor de str, que proporciona la información de calidad del aire correspondiente.

*cyberpi.cloud.time(option, location)*

-option: *str,* opción de tiempo

Rango de ajuste:

*sunrise\_time*: hora del amanecer

*sunrise\_hour*: hora del amanecer

*sunrise\_minute*: minuto de la hora del amanecer

*sunrise\_set*: hora de la puesta del sol

*sunset\_hour:* hora de la puesta del sol

*sunset\_minute:* minuto de la puesta del sol  
  
**A ser implementados**

*hour*: hora de la zona horaria especificada

*minute*: minuto de la zona horaria especificada

*second*: segundo de la zona horaria especificada

*year*: año de la zona horaria especificada

*month*: mes de la zona horaria especificada

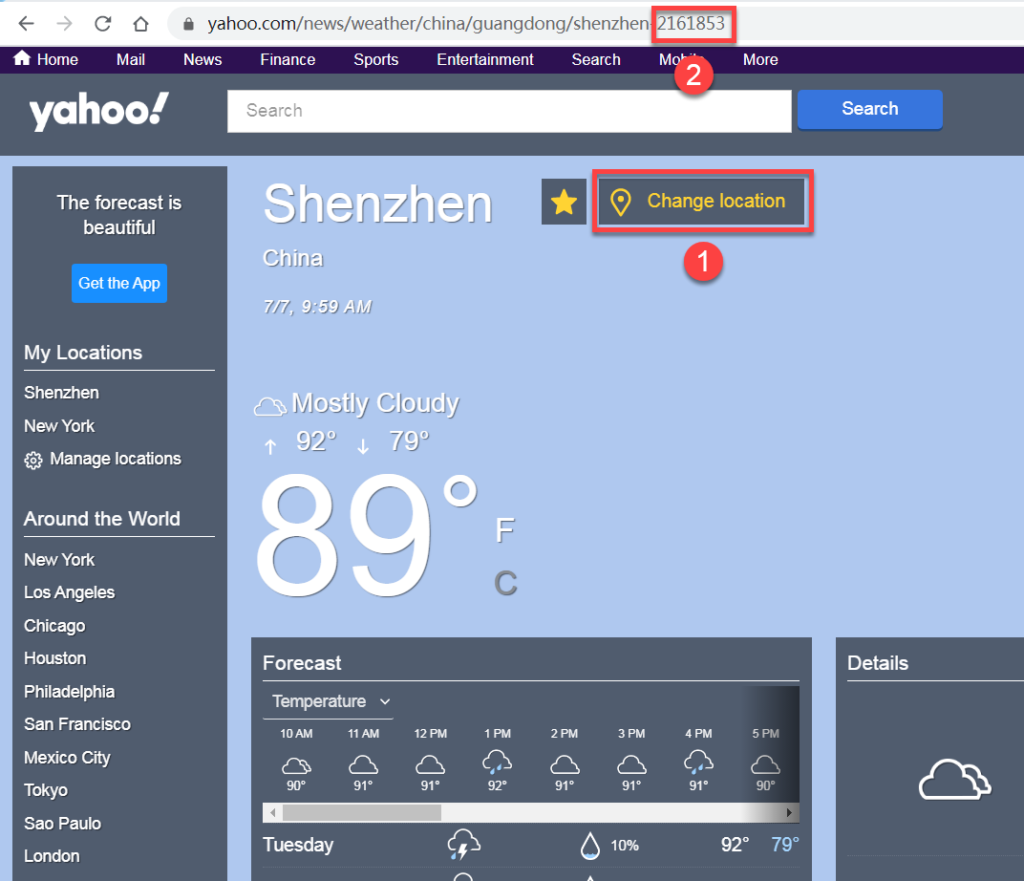
*day*: día de la semana de la zona horaria especificada

*date*: fecha de la zona horaria especificada

-location: *str* o *int*

*str*: código de la ciudad para la que se van a obtener los datos meteorológicos

Puedes visitar Yahoo Weather para obtener los códigos de las ciudades.



Pueden cambiar la ubicación a aquella en la que se buscará la información meteorológica y luego vean el código en la barra de direcciones.

Alternativamente, puede ver el código de una ciudad en el bloque correspondiente en mBlock 5, como se muestra en la siguiente figura.

**[A implementar]**  
*int*: zona horaria de la cual se desea obtener la información horaria.

Rango de ajuste: –12 a +12

Se devuelve un *str* o *int*value, que proporciona la información de tiempo correspondiente.

AI

CyberPi proporciona algunas capacidades de IA. Después de estar conectado a Internet, CyberPi puede reconocer discursos, leer textos en voz alta y traducir algunos textos a través de las interfaces de IA correspondientes.

*cyberpi.cloud.listen(language, t=3)*

Comienza a reconocer un discurso en el idioma especificado

Esta API bloquea el hilo hasta que se reconoce el discurso.  
  
Parámetros:

-language: *str*, idioma del habla a reconocer.

Rango de ajuste:

"chinese": Chino

"cantonese": Cantonés

"japanese": Japonés

"english": Inglés

"french": Francés

"german": Alemán

"spanish": Español

"portuguese": Portugués

"russian": Ruso

"korean": Koreano

"italian": Italiano

"dutch": Holandés.

-t: *float o str*; el valor predeterminado es 3.

Cuando se configura t en *int*, se indica el período en el que se reconoce un discurso (en segundos); y cuando lo configura en *str*, se puede configurar solo para grabar, lo que nos indica que el último archivo de audio grabado se usa para el reconocimiento de voz.

*cyberpi.cloud.listen\_result()*

Obtiene el último discurso reconocido

Se devuelve un valor *str*.

*cyberpi.cloud.tts( language, message)*

Lee el texto especificado en voz alta

Parámetros:

-language: *str*, idioma en el que se leerá el texto en voz alta.

Actualmente, el parámetro de idioma no se puede configurar debido a la limitación de los servicios en la nube. Esta API reconoce automáticamente el contenido del mensaje y lo lee en voz alta en chino o inglés.

*cyberpi.cloud.translate( language, message)*

Traduce un texto al idioma especificado.

-language: *str*, idioma del habla a reconocer.

Rango de ajuste:

"chinese": Chino

"cantonese": Cantonés

"japanese": Japonés

"english": Inglés

"french": Francés

"german": Alemán

"spanish": Español

"portuguese": Portugués

"russian": Ruso

"korean": Koreano

"italian": Italiano

"dutch": Holandés.

-message:*str,* texto a ser mostrado.

Transmisión

CyberPi admite múltiples tipos de transmisiones. Con estas transmisiones, CyberPi puede implementar la comunicación entre subprocesos y en LAN e Internet.

**Transmisión básica**

*cyberpi.broadcast(message)*Es para enviar un mensaje de difusión.

El mensaje puede ser recibido por todos los subprocesos en CyberPi o por sprites de mBlock en modo en vivo.

Parámetro:

-mesage: *str*, nombre del mensaje de difusión.

Transmisión LAN.

**Nota:**

Todos los CyberPis en la misma LAN deben usar el mismo canal de comunicación.

Sin conexión de enrutador, la cyberpi usa el mismo canal predeterminado y, por lo tanto, pueden comunicarse entre sí en la LAN; cuando CyberPi está conectado a un enrutador, su canal depende de la configuración del enrutador, que puede ser diferente del canal predeterminado y, por lo tanto, es posible que no se comunique con otro CyberPi que no esté conectado al enrutador.

Por lo tanto, para garantizar una comunicación adecuada, si usa un router, conecte todos los CyberPis en la misma LAN al router.

*cyberpi.wifi\_broadcast.set(message, val)*

Envía un mensaje de difusión LAN

Parámetros:

-message: *str*, nombre del mensaje de difusión LAN

-val: *str*, valor transmitido por el mensaje de difusión LAN

*cyberpi.wifi\_broadcast.get(message)*

Obtiene el valor transmitido por un mensaje de difusión LAN

Parámetro:

-message: *str*, nombre del mensaje de difusión LAN.

Se devuelve un valor *str.*

**Sugerencias:** es posible que CyberPi no reciba una transmisión cuando usa esta API en un programa. Algunas API pueden bloquear el hilo cuando se ejecutan, lo que hace que CyberPi pierda una transmisión.

El siguiente programa es un ejemplo: un CyberPi envía una transmisión cuando se ejecuta el código time.sleep(1), y el CyberPi que se supone que debe recibir la transmisión la pierde.  
  
*import cyberpi  
import time*

*while True:*

*cyberpi.led.on(0,0,255,1)*

*time.sleep(1)  
 cyberpi.wifi\_broadcast.get(message)*

Transmisión en modo de carga

*cyberpi.upload\_broadcast.set(message, val)*

Envía un mensaje de difusión en la nube del usuario

Para usar esta API, debe usar *wifi.is\_connect()* y *cloud.setkey(key)* para obtener el permiso para usar los servicios en la nube proporcionados por Makeblock para su CyberPi.

Los mensajes en la nube del usuario se pueden usar para compartir datos entre dispositivos y proyectos. Por ejemplo, después de iniciar sesión en mBlock 5 en varias PC, puede enviar un mensaje de difusión en la nube de usuario en una PC a todos los demás dispositivos de hardware en línea y sprites de mBlock 5 en las otras PC.

Con esta API, puede acceder y controlar fácilmente sus dispositivos de hardware a través de Internet. Además, puede usar mBlock 5 o mBlock-Python Editor para diseñar interfaces de usuario de interacción visual.

Parámetros:

-mensaje: *str,* nombre del mensaje de transmisión en la nube del usuario

-val: *str,* valor transmitido por el mensaje de difusión en la nube del usuario

*cyberpi.upload\_broadcast.set(message)*

Obtiene el valor transmitido por el mensaje de difusión en la nube del usuario especificado

Parámetro:

-mensaje: *str*, nombre del mensaje de transmisión en la nube del usuario

Se devuelve un valor *str.*

**Sugerencias:** es posible que CyberPi no reciba una transmisión cuando usa esta API en un programa. Algunas API pueden bloquear el hilo cuando se ejecutan, lo que hace que CyberPi pierda una transmisión.

El siguiente programa es un ejemplo: un CyberPi envía una transmisión cuando se ejecuta el código time.sleep(1), y el CyberPi que se supone que debe recibir la transmisión la pierde.

*import cyberpi*

*import time*

*while True:*

*cyberpi.led.on(0,0,255,1)*

*time.sleep(1)  
 cyberpi.cloud\_broadcast.get(message)*

Control de secuencias de comandos

*cyberpi.stop\_all()*

Detiene todos los scripts

*cyberpi.stop\_this()*

Detiene el script actual

*cyberpi.stop\_other()*

Detiene todos los scripts excepto el actual

*cyberpi.restart()*

Resetea la cyberpi.

Eventos

CyberPi admite la ejecución de múltiples subprocesos. Puede agregar un hilo para CyberPi agregando el encabezado de evento correspondiente y el hilo escucha el evento especificado. Cuando ocurre el evento, se ejecuta el código subsiguiente en el hilo.

**Nota:** Las API de eventos no se pueden usar en la programación de Python3. Debe usar el mecanismo de subprocesos proporcionado por Python3 para compilar código de subprocesos múltiples.

Compilación de eventos.

Los encabezados de eventos se compilan mediante decoradores, como se muestra en el siguiente ejemplo:

*import cyberpi*

*@cyberpi.event.start # Event to be listened to*

*def callback():*

*cyberpi.led.on("green") # Código que se ejecutará después de que ocurra el evento*

*@cyberpi.event.received('hello') # Evento (con un parámetro) a escuchar*

*def callback():*

*cyberpi.led.on("blue") # Código que se ejecutará después de que ocurra el evento*

De esta forma de codificación, cuando ocurre el evento definido en el encabezado del evento, se ejecuta el código posterior. En el ejemplo anterior, se usa el nombre de función *callback()*. De hecho, sin embargo, se puede usar cualquier nombre de función que cumpla con la convención de nomenclatura. Además, está bien usar nombres de funciones repetidamente porque las funciones se definen después de los eventos.

Como puede ver en el ejemplo anterior, la función *callback()* se ejecuta cuando inicia CyberPi y la otra función *callback()* se ejecuta cuando CyberPi recibe el mensaje de difusión "hola". No se produce ningún conflicto porque se definen después de diferentes eventos.

Mecanismo de determinación de eventos

Un encabezado de evento se activa solo cuando ocurre el evento escuchado. Por ejemplo, si se escucha el evento "botón x presionado", el encabezado del evento se activa y el código posterior se ejecuta solo cuando el estado del botón x cambia de "no presionado" a "presionado". Si el botón x se mantiene en estado presionado, el código subsiguiente no se ejecuta.

API de encabezado de evento para CyberPi

*cyberpi.event.start*Escucha el inicio de CyberPi y ejecuta el código posterior cuando se inicia CyberPi.

*cyberpi.event.is\_press(name)*Escucha el estado del botón especificado o la dirección de movimiento del joystick y ejecuta el código subsiguiente cuando se presiona el botón especificado o el joystick se mueve en la dirección especificada

Parámetro:

-name: *str*, nombre del botón o dirección del joystick

Rango de ajuste:

*a: Boton A (cuadrado)*

*b: Boton B (triangulo)*

*up:* joystick se mueve hacia arriba

*down*: joystick se mueve hacia abajo

*left:* joystick se mueve a la izquierda

*right:* joystick se mueve a la derecha

*middle*: joystick se presiona en el centro

*any\_direction:* joystick se mueve en cualquier dirección

*any\_button*: botón A o B

*any*: cualquier boton o direciión del joystick

un valor entero es devuelto

*cyberpi.event.is\_tiltforward*

Escucha la postura de CyberPi y ejecuta el siguiente código cuando CyberPi se inclina hacia adelante

*cyberpi.event.is\_tiltback*

Escucha la postura de CyberPi y ejecuta el siguiente código cuando CyberPi se inclina hacia atrás

*cyberpi.event.is\_tiltleft*

Escucha la postura de CyberPi y ejecuta el siguiente código cuando CyberPi se inclina hacia la izquierda

*cyberpi.event.is\_tiltright*

Escucha la postura de CyberPi y ejecuta el siguiente código cuando CyberPi se inclina hacia la derecha

*cyberpi.event.is\_faceup*

Escucha la postura de CyberPi y ejecuta el siguiente código cuando la pantalla de CyberPi mira hacia arriba

*cyberpi.event.is\_facedown*

Escucha la postura de CyberPi y ejecuta el siguiente código cuando la pantalla de CyberPi mira hacia abajo

*cyberpi.event.is\_shake*

Escucha el movimiento de CyberPi y ejecuta el siguiente código cuando se agita CyberPi

*cyberpi.event.is\_waveup*

Escucha el movimiento de CyberPi y ejecuta el siguiente código cuando se agita CyberPi

*cyberpi.event.is\_wavedown*

Escucha el movimiento de CyberPi y ejecuta el siguiente código cuando se agita CyberPi

*cyberpi.event.is\_waveleft*

Escucha el movimiento de CyberPi y ejecuta el siguiente código cuando se mueve CyberPi hacia la izquierda

*cyberpi.event.is\_waveright*

Listens to the motion of CyberPi and executes the subsequent code when CyberPi is waved to the right

*cyberpi.event.is\_freefall*

Escucha el movimiento de CyberPi y ejecuta el siguiente código cuando CyberPi se cae

*cyberpi.event.is\_clockwise*

Escucha el movimiento de CyberPi y ejecuta el código siguiente cuando CyberPi gira en el sentido de las agujas del reloj alrededor del eje z

*cyberpi.event.is\_anticlockwise*

Escucha el movimiento de CyberPi y ejecuta el código siguiente cuando CyberPi gira en sentido contrario a las agujas del reloj alrededor del eje z

*cyberpi.event.greater\_than(threshold, type)*

Escucha el resultado de la comparación y ejecuta el código posterior cuando el valor es mayor que el umbral

Parámetros:

-threshold: *int,* umbral; rango de ajuste: 0-100

-type: str, tipo de la fuente de datos

Rango de ajuste:

-microphone: micrófono

-light\_sensor: sensor de luz

-shake\_val: sacudiendo la fuerza

-timer: temporizador

*cyberpi.event.smaller\_than(threshold, type)*

Escucha el resultado de la comparación y ejecuta el código posterior cuando el valor es menor que el umbral.

Parámetros:

-threshold: *int,* umbral; rango de ajuste: 0-100

-type: str, tipo de la fuente de datos

Rango de ajuste:

-microphone: micrófono

-light\_sensor: sensor de luz

-shake\_val: sacudiendo la fuerza

-timer: temporizador

*cyberpi.event.receive(message)*

Escucha el mensaje de difusión especificado y ejecuta el código subsiguiente cuando se recibe el mensaje de difusión

Parámetro

-message: *str,* nombre del mensaje de difusión escuchado

*cyberpi.event.upload\_broadcast(message)*

Escucha el mensaje de transmisión del modo de carga especificado y ejecuta el código posterior cuando se recibe el mensaje de transmisión del modo de carga

Parámetro:

-message: *str*, nombre del mensaje de transmisión del modo de carga escuchado

*cyberpi.event.cloud\_broadcast(message)*

Escucha el mensaje de difusión en la nube del usuario especificado y ejecuta el código posterior cuando se recibe el mensaje de difusión en la nube del usuario

Parámetro:

message: *str,* nombre del usuario mensaje de difusión en la nube escuchado

*cyberpi.event.wifi\_broadcast(message)*

Escucha el mensaje de difusión de LAN especificado y ejecuta el código posterior cuando se recibe el mensaje de difusión de LAN

Parámetro:

-message: *str*, nombre del mensaje de difusión de LAN escuchado

prueba 15/07/2023 21:49 correo eden  
  
prueba 2 21:51 correo tamara