Módulo5 Cyberpi, mBot y Python

El software *mBlock* permite realizar la transición de la programación por bloques a lenguaje de texto Python. El proposito de este módulo de ampliación es ayudar al profesorado quiere iniciar a su alumnado en Python.

- Python
- Hola Mundo
- Audio
- LED
- Entradas
- Display
- Sensores de movimiento
- Sensor de sonidos
- Evita obstáculos
- Sensor de líneas
- Sigue líneas

Copyright 2026

Python

Librería cyberpi

Los programas en python de cyberpi que vamos a realizar utilizan esta librería por lo tanto hay que cargarla previamente en nuestros programas.

```
import cyberpi
```

Tiene multitud de funciones, te aconsejamos ver un vistazo en esta documentación (en inglés) :

https://www.lbotics.at/images/mbot2/files/mBot2_API_cyberpi.pdf

> Esta es **una muy breve introducción al Python** como recordatorio de algunas instrucciones si ya has utilizado este lenguaje.
>
> Si es la primera vez, te recomendamos que visites nuestro curso **PYTHON PARA TODOS**
>
> **Python for everybody** por Charles R. Severance licencia CC-BY-NCSA que empieza desde cero.

Lenguajes, intérpretes y compiladores

Python es un lenguaje **de alto nivel** destinado a ser relativamente sencillo para que los humanos lean y escriban y para que los ordenadores lean y procesen. Otros lenguajes de alto nivel incluyen Java, C ++, PHP, Ruby, Basic, Perl, JavaScript y muchos más. El hardware real dentro de la Unidad Central de Procesamiento (CPU) no comprende ninguno de estos lenguajes de alto nivel.

La CPU entiende un idioma que llamamos **lenguaje de máquina**. El lenguaje de máquina es muy simple y francamente muy tedioso de escribir porque está representado en ceros y unos:

El lenguaje de máquina parece bastante simple en la superficie, dado que solo hay ceros y unos, pero su sintaxis es aún más compleja y mucho más compleja que Python. Muy pocos programadores escriben lenguaje de máquina. En su lugar, creamos varios traductores para permitir que los programadores escriban en lenguajes de alto nivel como Python o JavaScript y estos traductores convierten los programas al lenguaje de máquina para su ejecución real por parte de la CPU.

Copyright 2026

Estos traductores de lenguaje de programación se dividen en dos categorías generales: (1) intérpretes y (2) compiladores.

Un **intérprete** lee el código fuente del programa como está escrito por el programador, analiza el código fuente e interpreta las instrucciones sobre la marcha. Python es un intérprete y cuando ejecutamos Python de forma interactiva, podemos escribir una línea de Python (una oración) y Python la procesa de inmediato y está lista para que escribamos otra línea de Python.

```
>>> x = 6
>>> print(x)
6
>>> y = x * 7
>>> print(y)
42
>>>
```

Está en la naturaleza de un **intérprete** poder tener una conversación interactiva como se muestra arriba. A un **compilador** debemos entregarle todo el programa en un archivo, y luego ejecuta un proceso para traducir el código fuente de alto nivel al lenguaje de máquina y luego el compilador coloca el lenguaje de máquina resultante en un archivo para su posterior ejecución.

Variables

Las variables son como cajas que puedes meter valores. Y los valores pueden ser de varios **tipos** :

- **int** si son enteros
- **float** si tienen decimales
- **binario**Deben comenzar por 0b. Por ejemplo: 0b110, 0b11
- **string** son frases, son "***cadenas***" de caracteres entre "
- **bool** Solamente hay dos literales booleanos True o False
- **lista** Se pueden declarar variables que son conjuntos por ejemplo Colores = ["verde", "rojo", "naranja"]

Para crear una variable puedes usar cualquier palabra, x, y, z o Nombre_alumno ... pero algunas palabras no puedes usar, ver

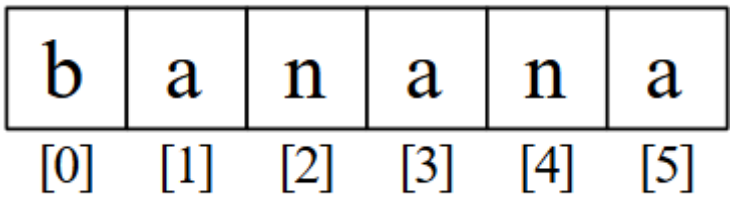
Para visualizar variables puedes usar la **instrucción print** poniendo entre paréntesis el valor o variable que quieres visualizar.

> Modifica los valores como quieras, es un **intérprete**, juega y dale al play para ver el resultado

Como puedes ver se ha introducido un operador el + que realiza la suma del valor de x original (43) y se le incrementa una unidad resultando en la impresión un 44.

Cadenas

Cadenas son secuencias de caracteres, por ejemplo la palabra "banana"



fuente 'Python for Everybody' por Charles R. Severance

Se puede obtener su longitud con la función len, o obtener un carácter ...

Operadores

Este apartado de operadores es adaptado de Federico Coca Guia de Trabajo de Microbit CC-BY-SA

Los **operadores aritméticos** se utilizan para realizar operaciones matemáticas como sumas, restas, multiplicaciones, etc.

Operador	Descripción	Ejemplo
+	Suma o concatenación en textos	`5+3=8`, `"Hola" + "Mundo" = "Hola Mundo"`
-	Diferencia	`6-3=3`
*	Multiplicación	`3*3=9`
/	División	`6/2=3`
//	Parte entera de un cociente	`10//3=3`
%	Resto de un cociente	`10%3=1`
**	Potenciación	`5**2=25`

Los **operadores de asignación** se utilizan para asignar valores a variables.

Operador	Descripción	Ejemplo
=	Asignación	`x=4`, `a = a + 1`
+=	Suma y asignación	`x+=1` equivale a `x = x + 1`
-=	Diferencia y asignación	`x-=1` equivale a `x = x - 1`
*=	Multiplicación y asignación	`x*=3` equivale a `x = x * 3`
/=	División y asignación	`x/=3` equivale a `x = x / 3`
%=	Asignación de restos	`x%=3` equivale a `x = x % 3`
**=	Asignación de exponentes	`x**=3` equivale a `x = x ** 3`

Los **operadores de comparación** comparan dos valores/variables y devuelven un resultado booleano: Verdadero o Falso `True` o `False`.

Operador	Descripción	Ejemplo
==	Igual a	`2==3` retorna `False`
!=	Distinto de	`2!=3` retorna `True`
<	Menor que	`2<3` retorna `True`
>	Mayor que	`2>3` retorna `False`
<=	Menor o igual que	`2<=3` retorna `True`
>=	Mayor o igual que	`2>=3` retorna `False`

Los **operadores lógicos** se utilizan para comprobar si una expresión es Verdadera o Falsa. Se utilizan en la toma de decisiones.

Operador	Descripción	Ejemplo
and	AND lógica	`a and b #True si a y b son ciertos`
or	OR lógica	`a or b #True si a o b son ciertos`
not	NOT lógica	`not a #True si el operador a es falso`
in	pertenencia	Devuelve True si pertenece
no int	no pertenencia	Devuelve True si no pertenece
is	identidad	Devuelve True si son iguales

Copyright 2026

Operador	Descripción	Ejemplo
is not	no identidad	Devuelve True si no son inguales

Los **operadores bit a bit** o bitwise actúan sobre los operandos como si fueran cadenas de dígitos binarios. Operan bit a bit:

Operador	Descripción	Ejemplo
&	AND bit a bit	`5&6 # 101 & 110 = 110 = 4`
|	OR bit a bit	`5 | 6 # 101 | 110 = 111 = 7`
~	NOT bit a bit	`~3 # ~011 = 100 = -4`
^	XOR bit a bit	`5^3 # 101^011 = 110 = 6`
<<	Desplazamiento izquierda	`4<<1 # 100 << 1 = 1000 = 8`
>>	Desplazamiento derecha	`4 >> 1 # 100 >> 1 = 010 = 2`

Comentarios en Python
Una sola línea : Escribiendo el símbolo almohadilla (#) delante del comentario.
Multilínea: Escribiendo triple comillas dobles (“””) al principio y al final del comentario.

Entradas de teclado

Ya hemos visto salidas por pantalla con **print**, pero ahora con input puede leer variables del teclado.
Fíjate que hay que poner las líneas **x = float (x)** e **y = float(y)** para convertirlos a números decimales, en caso contrario las interpreta string y no puede multiplicar en Resultado, pero en el siguiente ejemplo **no es necesario en la variable cel** (celsius) pues se multiplica por números decimales 32.0 5.0 y 9.0

try y **except** son dos funciones que son *un seguro para el programador* por si el usuario en vez de teclear un número, mete un string o carácter

La sangría es importante en Python
La sangría se refiere a los espacios al comienzo de una línea de código. Mientras que en otros lenguajes de programación la sangría en el código es solo para facilitar la lectura, la sangría en Python es muy importante ya que se usa para indicar un bloque de código.

Copyright 2026

Condicionales

Las instrucciones **if: else:** son las que nos permiten realizar operaciones según las condiciones puestas. *Ojo con la sangría*

> **\n** es un carácter especial que significa "Salto de página"

Bucles

- **while** ejecuta lo contenido en la sangría mientras sea verdadero la condición
- **for** ejecuta lo contenido en la sangría mientras y va recorriendo la variable dentro del rango creado

Para verlo mejor vamos a ver estos ejemplos

- EJEMPLO BUCLE WHILE
  - mientras n sea positivo va ejecutando : imprime n y lo decrementa
  - al decrementar llega un momento que deja de ser positivo y finaliza el bucle
- EJEMPLO BUCLE WHILE INFINITO
  - Es muy típico en robótica, todo el rato hace el bucle (en robótica para que lea los sensores y realice cosas en los actuadores) pero este ejemplo no esta en un robot sino en tu pc y no queremos que se quede "colgado" luego al teclear "fin" acaba gracias a la instrucción **break**
  - Fíjate que hay una instrucción **continue** para que pase a la siguiente iteración provocando que no imprime lo tecleado
- EJEMPLO BUCLE FOR FRIENDS
  - Va recorriendo la variable friend dentro del cojunto lista friends
  - como puedes ver la diferencia entre for y while es que for además recorre la variable
- EJEMPLO BUCLE FOR
  - mientras n este en el rango de 0 a 5 se ejecuta

Funciones

No vamos a entrar en detalle, pero observa el siguiente código

- **FUNCIONES PREDEFINIDAS** Si observas, la primera línea llama a importar una librería externa, **import math** donde math es un fichero que tienen funciones predefinidas, vamos a utilizar una de ellas, la raiz cuadrada **sqrt** luego para llamar a esa función que esta definida dentro de math se hace con la instrucción **math.sqrt**
- **FUNCIONES DEFINIDAS POR TI** em este caso, se utiliza la palabra **def** para crear una función, que le vamos a pasar tres argumentos a, b y c y para finalizar la función usamos

Copyright 2026

return para devolver el valor que queremos obtener

Para saber más de Python

CURSO PYTHON FOR EVERYBODY en español	ver
Curso completo de Python 222pag pdf (*)	Descargar
Curso completo de Python 422pag (*)	Descargar
Curso completo de Python desde 0 (*)	Ver
Curso de Python desde 0 (*)	Ver
Manual de referencia Python (*)	Ver
Programación en Python (*)	Ver
Trabajando con ficheros en Python (*)	Ver
Programación orientada a objeto en Python (*)	Ver
un manual para aquellos usuarios con previos conocimientos de Python, como la programación modular y orientada a objetos. También algunos conocimientos de las librerías **tkinter** (Para crear interfaces gráficos y **SQlite3** (para gestionar bases de datos). (*)	Descargar

(*) Agradecimientos a Pere Manel http://peremanelv.com

Copyright 2026

Hola Mundo

Nuestro primer programa es muy sencillo: Que por el display del Cyberpi salga la típica frase del primer programa principiante

A la hora de utilizar las funciones de la librería cyberpi, tienes dos opciones:

- Programar sin omitir de donde viene (o sea, indicar el objeto de donde sale esa función, **cyberpi.**)
- Programar omitiéndolo pero antes tienes que decirle que importas todo **import** *

Mi primer programa Hola Mundo (método sin omitir)

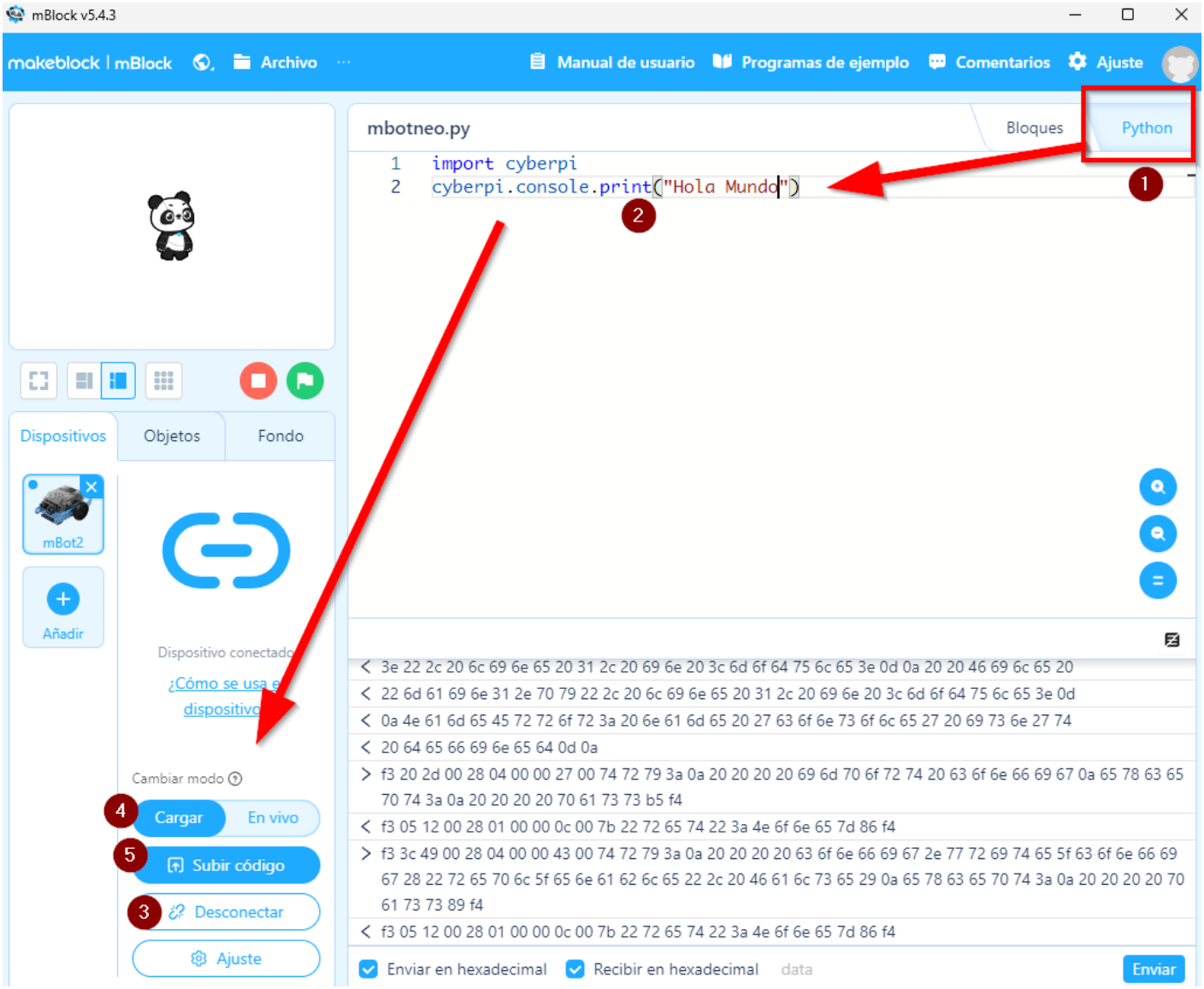
Entramos en mBlock, y en la pestaña de Python tecleamos este programa:

```
import cyberpi
cyberpi.console.print("Hola Mundo")
```

> Previamente tienes que tener el programa **mBlock** instalado y cargado el dispositivo **mBot2** (aunque para estos ejemplos con Cyberpi también vale) , recuerda lo visto en https://libros.catedu.es/books/cyberpi-y-mbot2/page/mblock5 y en https://libros.catedu.es/books/cyberpi-y-mbot2/page/como-usar-mbot2-en-mblock

1. Entramos en mblock con el dispositivo cargado mBot2 y vamos a la pesaña Python
2. Pegamos el programa
3. Vamos a conectar mBot2 (debemos de tenerlo conectado con un cable USB tal y como vimos en https://libros.catedu.es/books/cyberpi-y-mbot2/page/mi-primer-programa-hola-mundo
4. Damos a la opción Cargar
5. Subir código

Copyright 2026



Y el resultado es



Mi primer programa Hola Mundo (método omitiendo)

Repite los pasos anteriores pero con este código

```
from cyberpi import *
console.print("Hola Mundo")
```

> ¿Ves la diferencia de código?

Audio

Un tono

Un programa sencillo de dar un tono puede ser el siguiente

```
import cyberpi

cyberpi.audio.set_vol(100)
cyberpi.audio.play_tone(700,1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

Instrumentos

También podemos reproducir instrumentos

```
import cyberpi

cyberpi.audio.set_vol(100)
# str type eg. snare,bass-drum,side-stick,crash-cymbal,open-hi-hat,close-hi-
hat,tambourine,hand-clap,claves
# float beat > 0 (second)

cyberpi.audio.play_drum("snare",1)
cyberpi.audio.play_drum("snare",1)
cyberpi.audio.play_drum("side-stick",1)
cyberpi.audio.play_drum("tambourine",1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

Efectos sonoros

También este código nos selecciona varios efectos sonoros y los reproduce

Copyright 2026

```
import cyberpi

sound_effect =
["hello","hi","bye","yeah","wow","laugh","hum","sad","sigh","annoyed","angry","surprised","yum
my","curious","embarrassed","ready","sprint","sleepy","meow","start","switch","beeps","buzzing
","explosion","jump","laser","level-up","low-energy","prompt-
tone","right","wrong","ring","score","wake","warning","metal-clash","shot","glass-
clink","inflator","running water","clockwork","click","current","switch","wood-
hit","iron","drop","bubble","wave","magic","spitfire","heartbeat","load"]

cyberpi.display.show_label('UP  :GO   UP\nDOWN:GO   DOWN\nMID :PLAY EFFECT', 16, 0, 0, 0)
cyberpi.display.show_label('SELECT:\nName:', 16, 0, 60, 1)

selected = 0
min_effect = 0
max_effect = len(sound_effect) - 1

while True:
    if cyberpi.controller.is_press('up'):
        if selected < max_effect:
            selected += 1
        else:
            selected = min_effect
    elif cyberpi.controller.is_press('down'):
        if selected > min_effect:
            selected -= 1
        else:
            selected = max_effect
    elif cyberpi.controller.is_press('middle'):
        cyberpi.led.on(255,0,0,id="all")
        cyberpi.audio.play_until(sound_effect[selected])
        cyberpi.led.on(0,0,0,id="all")

    cyberpi.display.show_label('{}'.format(selected), 16, 60, 60, 2)
    cyberpi.display.show_label('{}'.format(sound_effect[selected]), 12, 52, 80, 3)
```

Copyright 2026

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/2EPpDJqUhew

Grabadora

O hacernos una grabadora de bolsillo

```
import cyberpi
from time import sleep

cyberpi.audio.set_vol(100)

cyberpi.display.show_label("A:Start Recording\nB:Play Recording",12,0,0,0)

while True:
    cyberpi.display.show_label("Waiting.",16,0,40,1)
    if cyberpi.controller.is_press('a'):
        cyberpi.led.on(0,255,0,id="all")
        cyberpi.display.show_label("Listening..",16,0,40,1)
        cyberpi.audio.record()
        sleep(5)
        cyberpi.display.show_label("Finished..",16,0,40,1)
        cyberpi.audio.stop_record()

    elif cyberpi.controller.is_press('b'):
        cyberpi.led.on(0,0,255,id="all")
        cyberpi.display.show_label("Playing..",16,0,40,1)
        cyberpi.audio.play_record_until()
        cyberpi.display.show_label("Finished..",16,0,40,1)

    cyberpi.display.show_label("Waiting...",16,0,40,1)
    cyberpi.led.on(0,0,0,id="all")
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/ayyBMRfwBpQ

LED

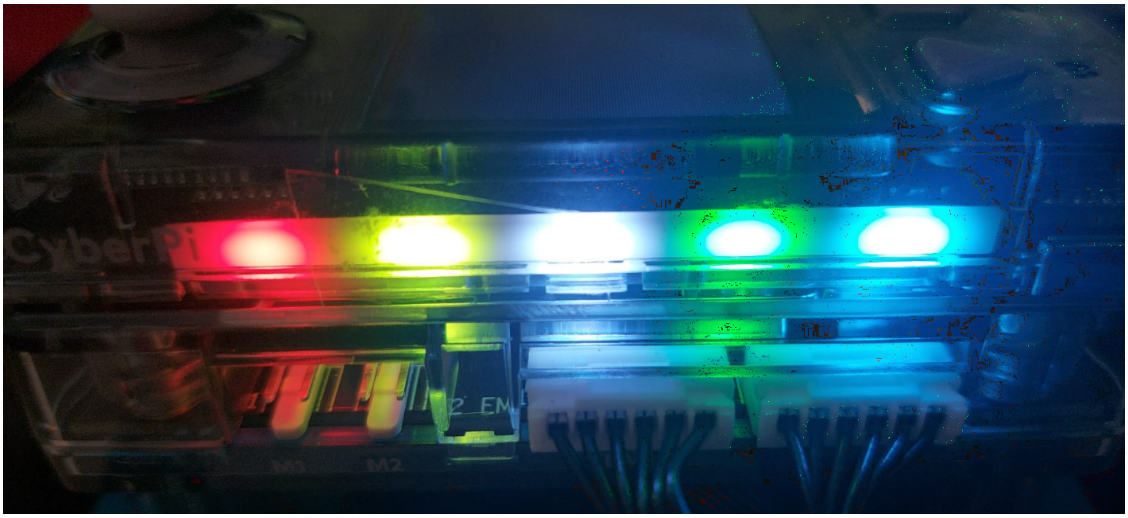
Colores

Podemos fijar los colores de los leds de Cyberpi con este código

```
import cyberpi

#               R  G B Position
cyberpi.led.on(255,0,0,id=1)
cyberpi.led.on(255,255,0,id=2)
cyberpi.led.on(255,255,255,id=3)
cyberpi.led.on(0,255,0,id=4)
cyberpi.led.on(0,255,255,id=5)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT



Intermitencia

Podemos encender y apagar a voluntad

```
import cyberpi
from time import sleep
```

```
while True:
    #               R  G B Position
    cyberpi.led.on(255,0,0,id=1)
    cyberpi.led.on(255,255,0,id=2)
    cyberpi.led.on(255,255,255,id=3)
    cyberpi.led.on(0,255,0,id=4)
    cyberpi.led.on(0,255,255,id=5)
    sleep(1)
    #               Position
    cyberpi.led.off(id='all')
    sleep(1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

Arco Iris

```
import cyberpi

"""
Name List (str)

rainbow , spoondrift , meteor_blue , meteor_green ,
flash_red , flash_orange , firefly

"""

cyberpi.led.play(name = "rainbow")
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/KrjRfHxDUwk

Movimiento leds

```
import cyberpi
from time import sleep

cyberpi.led.on(255,0,0,id=1)
while True:
    # shift all the colors 1 step to the right.
    # (If it is negative, it will be left shifting.)
    cyberpi.led.move(1)
    sleep(1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/oorI8BS_xZU

Entradas

Botones A y B

Si aprieto el botón A pues enciendo leds, si aprieto B los apago

```
import cyberpi

"""
Button Name List (str)

a , b
up, down, left , right , middle
any_direction , any_button , any

"""
while True:
                              # button name
    if cyberpi.controller.is_press('a'):
        cyberpi.led.on(255,0,0,id='all')
        cyberpi.console.println("LED ON!")
    if cyberpi.controller.is_press('b'):
        cyberpi.led.on(0,0,0,id='all')
        cyberpi.console.println("LED OFF!")
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/1w7q_3t6nEM

Intensidad de la luz

La función show_label la veremos después, tiene el formato cyberpi.display.show_label(texto, tamaño, color, x, y)

```
import cyberpi

cyberpi.display.show_label("LUZ=",16,0,0,0)

while True:
    CantidadLuz = cyberpi.get_brightness()
    cyberpi.display.show_label(CantidadLuz,16,0,60,1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/DyamGMRte7w

Cantidad de sonido

```
import cyberpi

cyberpi.display.show_label("Loudness:",16,10,10,index=1)

while True:
    loudness_value = cyberpi.get_loudness()
    cyberpi.display.show_label(loudness_value,16,80,10,index=2)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/_xFezHI2N9o

Nivel de batería

```
import cyberpi

cyberpi.display.show_label("Battery Level",16,10,0,index=0)
cyberpi.display.show_label("Builtin:",16,10,20,index=1)
cyberpi.display.show_label("Extra:",16,10,40,index=2)

while True:
    builtin_batt = cyberpi.get_battery()
    extra_batt = cyberpi.get_extra_battery()

    cyberpi.display.show_label(builtin_batt,16,80,20,index=3)
    cyberpi.display.show_label(extra_batt,16,80,40,index=4)
```

Copyright 2026

Display

Contador

```
import cyberpi

cyberpi.display.show_label("Counter Program",16,0,0,0)
counter = 0

while True:

    if counter < 100:
        counter = counter + 1
    else:
        counter = 0
    cyberpi.display.set_brush(counter+100, 0, 0)
    cyberpi.display.show_label(counter,32,48,64,1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia [MIT](MIT)

https://www.youtube.com/embed/yOK__g1O6OQ

Limpiar y apagar la pantalla

```
import cyberpi

cyberpi.display.on()

cyberpi.display.show_label("A:Clear the Sceen",12,0,0,0)
cyberpi.display.show_label("B:Close the screen",12,0,24,1)

while True:
```

```
    if cyberpi.controller.is_press('a'):
        cyberpi.display.clear()
    elif cyberpi.controller.is_press('b'):
        cyberpi.display.off()
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/qWmmkTRykyo

Linechart

El display permite visualizar gráficas como por ejemplo este código

```
import cyberpi

value = 0

while True:
    if value < 100:
        value = value + 1
    else:
        value = 0
    cyberpi.linechart.add(value)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/oSzWt8k3YLA

Barchart

```
import cyberpi
```

```
value = 0

while True:
    if value < 100:
        value = value + 0.1
    else:
        value = 0
    cyberpi.barchart.add(value)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/Y950W81RpEA

Table

```
import cyberpi

cyberpi.display.set_brush(255,0,0)
cyberpi.table.add(1,1,"1,1")
cyberpi.table.add(2,1,"2,1")
cyberpi.table.add(3,1,"3,1")
cyberpi.table.add(4,1,"4,1")

cyberpi.display.set_brush(0,255,0)
cyberpi.table.add(1,2,"1,2")
cyberpi.table.add(2,2,"2,2")
cyberpi.table.add(3,2,"3,2")
cyberpi.table.add(4,2,"4,2")

cyberpi.display.set_brush(0,0,255)
cyberpi.table.add(1,3,"1,3")
cyberpi.table.add(2,3,"2,3")
cyberpi.table.add(3,3,"3,3")
cyberpi.table.add(4,3,"4,4")
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia [MIT](MIT)



Drawpixel

```
import cyberpi
import time


pixel_list =
["music","picture","video","clock","play","pause","next","prev","sound","temperature","light",
"motion","home","gear","list","right","wrong","shut_down","refresh","trash_can","download","su
nny","cloudy","rain","snow","train","rocket","car","truck","droplet","distance","fire","magnet
ic","gas","vision","color","overcast","foggy","sandstorm"]

my_sprite = cyberpi.sprite()
for p in pixel_list:

    my_sprite.draw_pixel(p)
    my_sprite.set_size(200)
```

```
    time.sleep(1)
    cyberpi.screen.render()
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/0UBumFzEuvI

QR

```
import cyberpi

my_sprite = cyberpi.sprite()
my_sprite.draw_QR("https://catedu.es/")
my_sprite.set_size(400)
cyberpi.screen.render()
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT



¡¡ y funciona !!!

Copyright 2026

Sensores de movimiento

Sensor agitación

```
import cyberpi

cyberpi.display.show_label("Shake Value\nA:Start",16,0,0,0)

while not cyberpi.controller.is_press('a'):
    pass

while True:
    shake_value = cyberpi.get_shakeval()
    cyberpi.display.show_label("Shake Value",16,20,0,0)
    cyberpi.display.show_label("{}%".format(shake_value),24,40,50,1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/bTmeeCBABAA

Inclinación y rotación

El siguiente programa enseña los dos ángulos de inclinación en eje X e Y y rotación en eje Z

```
import cyberpi

cyberpi.display.show_label("YAW PITCH ROLL\nA:Start",16,0,0,0)

while not cyberpi.controller.is_press('a'):
    pass
while True:
```

```
    pitch = cyberpi.get_pitch()
    roll = cyberpi.get_roll()
    yaw = cyberpi.get_yaw()

    cyberpi.display.show_label("Yaw\n\nPitch\n\nRoll\n\n",16,0,0,0)
    cyberpi.display.show_label("{}\n\n{}\n\n{}".format(pitch,roll,yaw),16,50,0,1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

El resultado es muy parecido con la función de gyro

```
import cyberpi

cyberpi.reset_rotation(axis='all')
cyberpi.display.show_label("Gyroscope\nA:Start",16,0,0,0)

while not cyberpi.controller.is_press('a'):
    pass

while True:
    x_gyro = cyberpi.get_gyro('x')
    y_gyro = cyberpi.get_gyro('y')
    z_gyro = cyberpi.get_gyro('z')

    cyberpi.display.show_label("X\n\nY\n\nZ\n\n",16,0,0,0)
    cyberpi.display.show_label("{}\n\n{}\n\n{}".format(x_gyro,y_gyro,z_gyro),16,50,0,1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

Y rotation

```
import cyberpi

cyberpi.reset_rotation(axis='all')
cyberpi.display.show_label("Rotation\nA:Start",16,0,0,0)

while not cyberpi.controller.is_press('a'):
```

```
    pass

while True:
    x_rotate = cyberpi.get_rotation('x')
    y_rotate = cyberpi.get_rotation('y')
    z_rotate = cyberpi.get_rotation('z')

    cyberpi.display.show_label("X\n\nY\n\nZ\n\n",16,0,0,0)
    cyberpi.display.show_label("{}\n\n{}\n\n{}".format(x_rotate,y_rotate,z_rotate),16,50,0,1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

https://www.youtube.com/embed/i3HnE8lIdLo

Sensor de sonidos

Medición de distancia

```
import cyberpi,mbuild

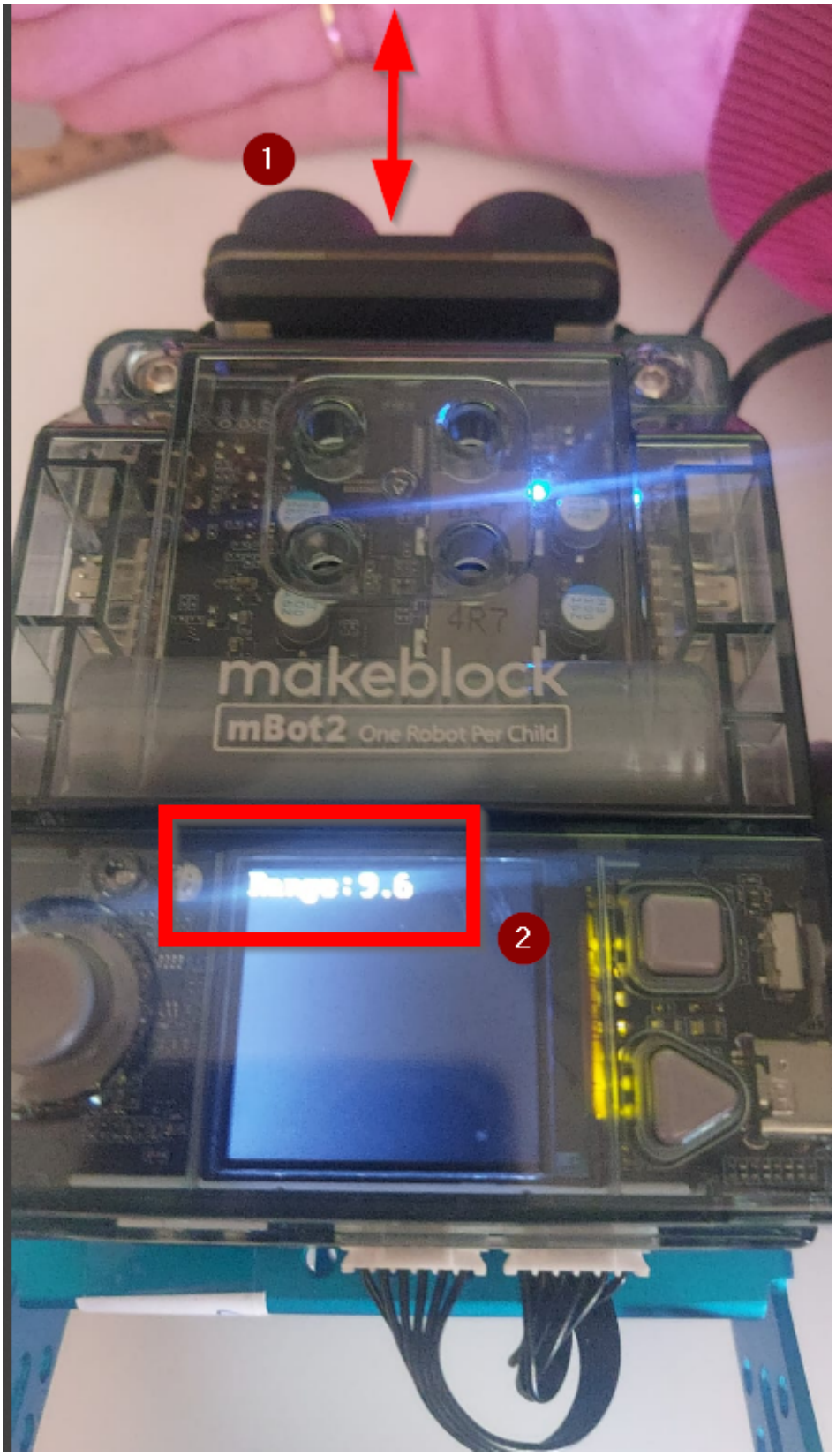

cyberpi.display.show_label("Range:", 16, 0, 0, index = 0)


while True:
    range = mbuild.ultrasonic2.get(index = 1)
    cyberpi.display.show_label(range, 16, 50, 0, index = 1)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

Podemos experimentar que es bastante preciso, más que en mBot1

Copyright 2026



Este código haría lo mismo pero alarmando que hay obstáculo o no

```
import cyberpi,mbuild

cyberpi.display.show_label("Range:", 16, 20, 0, index = 0)

while True:
    range = mbuild.ultrasonic2.get(index = 1)
    cyberpi.display.show_label(range, 16, 70, 0, index = 1)

    if range < 10:
        cyberpi.led.on(255,0,0,id="all")
        cyberpi.display.show_label("Obstacle!", 16, 0, 20, index = 2)
    else:
        cyberpi.led.on(0,255,0,id="all")
        cyberpi.display.show_label("No Obstacle", 16, 0, 20, index = 2)
```

Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT

Evita obstáculos

Basándonos en el código que hemos visto de https://github.com/PerfecXX/Python-mBot2/blob/main/example/mbuild/01-Ultrasonic%20Sensor2/02-Obstacle%20Detection.py le añadimos las instrucciones de

- Si hay obstáculo que gire
- Si no hay obstáculo que tire p'alante

Previamente una rutina de no empezar si no se aprieta el botón A

```
import cyberpi,mbuild, mbot2

cyberpi.display.show_label("PULSA A PARA COMENZAR",16,0,20,1)
while not cyberpi.controller.is_press('a'):
        pass

while True:
    range = mbuild.ultrasonic2.get(index = 1)
    cyberpi.display.show_label(range, 16, 70, 0, index = 1)

    if range < 40:
        cyberpi.led.on(255,0,0,id="all")
        cyberpi.display.show_label("Obstacle!", 16, 0, 20, index = 2)
        cyberpi.display.show_label("Girando un segundo",12,0,60,2)
        mbot2.turn_left(60,1)
    else:
        cyberpi.led.on(0,255,0,id="all")
        cyberpi.display.show_label("No Obstacle", 16, 0, 20, index = 2)
        mbot2.drive_power(70, -70)
```

Hace un poco el "*caballito*"", a lo mejor habría que quitarle un poco la potencia en mbot2.drive_power(70, -70) pero mola :

https://www.youtube.com/embed/H6CovqWfWvc

Copyright 2026

Sensor de líneas

Probando la detección de líneas.

```
import cyberpi,mbuild

cyberpi.display.show_label("RGB PROBE STATE\nL2:\nL1:\nR1:\nR2:\n", 16, 0, 0, index = 0)

while True:
    l2_line_state = mbuild.quad_rgb_sensor.is_line("L2",1)
    l1_line_state = mbuild.quad_rgb_sensor.is_line("L1",1)
    r1_line_state = mbuild.quad_rgb_sensor.is_line("R1",1)
    r2_line_state = mbuild.quad_rgb_sensor.is_line("R2",1)

    cyberpi.display.show_label(l2_line_state, 16, 30, 18, index = 1)
    cyberpi.display.show_label(l1_line_state, 16, 30, 36, index = 2)
    cyberpi.display.show_label(r1_line_state, 16, 30, 54, index = 3)
    cyberpi.display.show_label(r2_line_state, 16, 30, 72, index = 4)
```

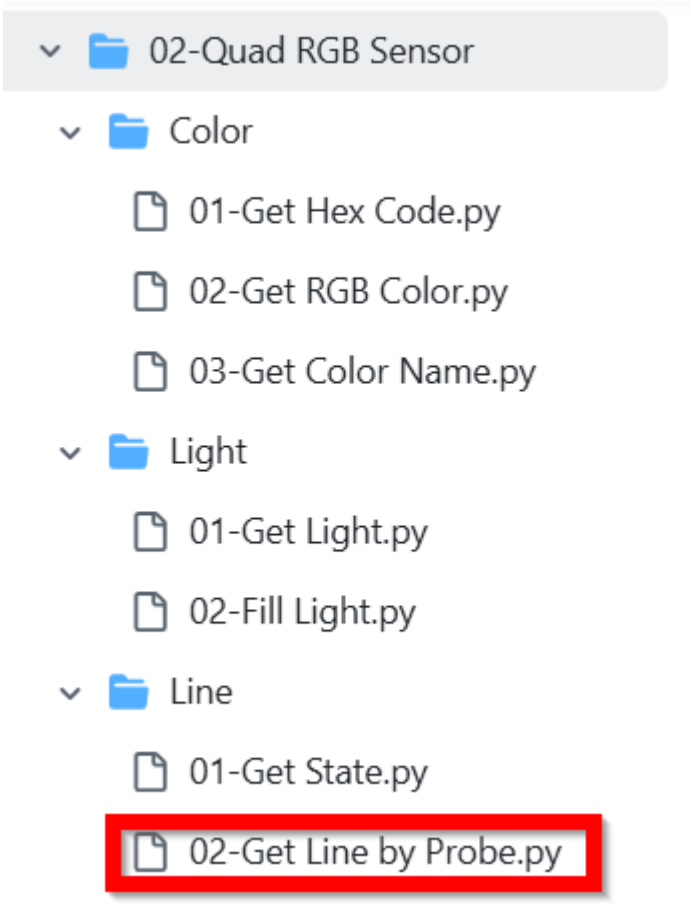
Extraído de https://github.com/PerfecXX/Python-mBot2/blob/main/README.md licencia MIT
Como puedes ver va detectando las líneas en los 4 sensores que tiene :

https://www.youtube.com/embed/6gyT5P1kMw8

Más probatinas...

En https://github.com/PerfecXX/Python-mBot2/tree/main/example/mbuild/02-Quad%20RGB%20Sensor puedes descargarte más scripts que visualizan como el sensor puede detectar el color y el brillo. Nosotros aquí sólo te hemos enseñado el último el de detectar la línea.

Copyright 2026



Copyright 2026

Sigue líneas

En este caso sólo vamos a utilizar de los 4 sensores quad que tiene los 2 de en medio con la instrucción

```
mbuild.quad_rgb_sensor.get_line_sta("middle", 1)
```

El valor que devuelve esta instrucción puede ser :

- **0 o en binario 00** que quiere decir que los dos sensores detectan blanco, por lo tanto habría que volver hacia **atrás** para recuperar la línea
- **1 o en binario 01** que significa que el sensor de la derecha detecta línea pero el de la izquierda no, por lo tanto hay que girar a la **derecha**
- **2 o en binario 10** que significa que el sensor de la izquierda detecta línea pero el de la derecha no, por lo tanto hay que girar a la **izquierda**
- **3 o en binario 11** que quiere decir que los dos sensores detectan línea, por lo tanto todo bien, **recto**

El código es

```
import event, time, cyberpi, mbuild, mbot2

cyberpi.display.show_label("PULSA A PARA COMENZAR",16,0,20,1)
while not cyberpi.controller.is_press('a'):
        pass

while True:
    if mbuild.quad_rgb_sensor.get_line_sta("middle", 1) == 0b00:
        mbot2.backward(50)

    if mbuild.quad_rgb_sensor.get_line_sta("middle", 1) == 0b11:
        mbot2.forward(50)

    if mbuild.quad_rgb_sensor.get_line_sta("middle", 1) == 0b01:
        mbot2.turn_right(50)
```

```
    if mbuild.quad_rgb_sensor.get_line_sta("middle", 1) == 0b10:
        mbot2.turn_left(50)
```

Fuente Javier Quintana

El resultado es

https://www.youtube.com/embed/6itHWcvZnUs