

Motion

APIs described in this section are provided for mBot2 to implement common motion control. Ensure that you have built mBot2 when you use these APIs. You can control the motion of mBot2 by using these APIs only when it is built with the 180 encoder motors, wheels, and tires delivered with it.



`mbot2.forward(speed = 50, t)`

Moves mBot2 forward at the specified speed

Parameters:

- *speed*: `float`, speed at which the encoder motors of mBot2 rotate, ranging from `-200` to `+200` RPM

If you set `speed` to a negative value, mBot2 moves backward.

- *t*: `float` or `str`

The default value is `"null"`.

When *t* is `float`, it indicates the duration mBot2 keeps moving forward, ranging from `0` to `+∞`, in seconds.

When *t* is `"null"`, it indicates that mBot2 keeps moving forward at the specified speed until it receives a command instructing it to stop moving forward.



```
mBot2.backward(speed = 50, t)
```

Moves mBot2 backward at the specified speed

Parameters:

- *speed*: `float`, speed at which the encoder motors of mBot2 rotate, ranging from `-200` to `+200` RPM

If you set `speed` to a negative value, mBot2 moves forward.

- *t*: `float` or `str`

The default value is `"null"`.

When *t* is `float`, it indicates the duration mBot2 keeps moving backward, ranging from `0` to `+∞`, in seconds.

When *t* is `"null"`, it indicates that mBot2 keeps moving backward at the specified speed until it receives a command instructing it to stop moving backward.



```
mBot2.turn_left(speed = 50, t)
```

Turns mBot2 left at the specified speed

Parameters:

- *speed*: `float`, speed at which the encoder motors of mBot2 rotate, ranging from `-200` to `+200` RPM

If you set `speed` to a negative value, mBot2 turns right.

- *t*: `float` or `str`

The default value is `"null"`.

When *t* is `float`, it indicates the duration mBot2 keeps turning left, ranging from `0` to `+∞`, in seconds.

When *t* is `"null"`, it indicates that mBot2 keeps turning left at the specified speed until it receives a command instructing it to stop turning left.



```
mBot2.turn_right(speed = 50, t)
```

Turns mBot2 right at the specified speed

Parameters:

- *speed*: `float`, speed at which the encoder motors of mBot2 rotate, ranging from `-200` to `+200` RPM

If you set `speed` to a negative value, mBot2 turns left.

- *t*: `float` or `str`

The default value is `"null"`.

When *t* is `float`, it indicates the duration mBot2 keeps turning right, ranging

from `0` to `+∞`, in seconds.

When `t` is `"null"`, it indicates that mBot2 keeps turning right at the specified speed until it receives a command instructing it to stop turning right.

MP**`mBot2.straight(distance, speed = 50)`**

Moves mBot2 forward the specified distance

To ensure the accuracy of the distance mBot2 moves, it moves at the preset speed when this API is executed. When this API is executed, mBot2 keeps moving forward until it moves the specified distance or another API is executed. For example, when `mBot2.stop()` is executed, mBot2 stops moving.

Parameters:

- *distance*: `float`, distance mBot2 is to move forward, in centimeters.
If you set `distance` to a negative value, mBot2 moves backward.
- *speed*: `float`, speed at which the encoder motors of mBot2 rotate, ranging from `0` to `200` RPM
The default value is 50. If you set `speed` to a negative value, the speed is the absolute value of the value you set.

MP**`mBot2.turn(angle, speed = 50)`**

Turns mBot2 clockwise the specified angle

To ensure the accuracy of the angle mBot2 turns, it moves at the preset speed when this API is executed. When this API is executed, mBot2 keeps turning clockwise until it turns the specified angle or another API is executed. For example, when `mBot2.stop()` is executed, mBot2 stops turning.

Parameters:

- *angle*: `int`, number of degrees mBot2 is to turn clockwise.
If you set `angle` to a negative value, mBot2 turns counterclockwise.
- *speed*: `float`, speed at which the encoder motors of mBot2 rotate, ranging from `0` to `+200` RPM
The default value is 50.

MP**`mBot2.drive_power(EM1_power, EM2_power)`**


Sets the power of the two encoder motors, EM1 (connected to the left wheel of mBot2 by default) and EM2 (connected to the right wheel by default)

This API is used to ensure the synchronization of the two encoder motors.

Parameters:

- **EM1_power:** `float` , power of the encoder motor EM1 (connected to the left wheel by default), ranging from `-100` to `+100` , in percentage
If you set this parameter to a positive value, the output shaft of EM1 rotates counterclockwise.
- **EM2_power:** `float` , power of the encoder motor EM2 (connected to the right wheel by default), ranging from `-100` to `+100` , in percentage
If you set this parameter to a positive value, the output shaft of EM2 rotates counterclockwise.

Example program

Python |  Copy

```

1  import cyberpi
2
3
4  speed = 50
5  cyberpi.audio.set_vol(10)
6  while True:
7      if cyberpi.controller.is_press('a'):
8          cyberpi.audio.play_until('switch')
9          cyberpi.mbot2.drive_power(speed, speed)
10     if cyberpi.controller.is_press("b"):
11         cyberpi.audio.play_until('switch')
12         cyberpi.mbot2.drive_power(0, 0)

```



mbot2.drive_speed(EM1_speed, EM2_speed)

Sets the rotating speed of the two encoder motors, EM1 (connected to the left wheel of mBot2 by default) and EM2 (connected to the right wheel by default)

This API is used to ensure the synchronization of the two encoder motors.

Parameters:

- **EM1_speed:** `float` , rotating speed of the encoder motor EM1 (connected to the left wheel by default), ranging from `-200` to `+200` RPM
If you set this parameter to a positive value, the output shaft of EM1 rotates counterclockwise.
- **EM2_speed:** `float` , rotating speed of the encoder motor EM2 (connected to the right wheel by default), ranging from `-200` to `+200` RPM
If you set this parameter to a positive value, the output shaft of EM2 rotates counterclockwise.



```
mbot2.EM_stop(port = "all")
```

Stops the rotating of the encoder motor connected to the specified port

Parameter:

port: `float` or `str`, port to which the encoder motor is connected

The default value is `"all"`. You can set this parameter to one of the following values:

`"all"`

`"em1"`

`"em2"`

`"ALL"`

`"EM1"`

`"EM2"`

`1`

`2`

Driving encoder motors

mBot2 is equipped with two encoder motors, which provide strong and accurate power output for mBot2. Note that the rotating speed of the encoder motors may be lower than the one you set if their load is too heavy.



```
mbot2.EM_set_power(power, port)
```

Sets the power of the encoder motor connected to the specified port

Parameters:

- *power*: `float`, power of the encoder motor EM1 (connected to the left wheel by default), ranging from `-100` to `+100`, in percentage

If you set this parameter to a positive value, the output shaft of EM1 rotates counterclockwise.

- *port*: `float` or `str`, port to which the encoder motor is connected

The default value is `"all"`. You can set this parameter to one of the following values:

`"all"`

`"em1"`

`"em2"`

`"ALL "``"EM1 "``"EM2 "``1``2`**MP**`mbot2.EM_set_speed(speed, port)`

Sets the rotating speed of the encoder motor connected to the specified port

Parameters:

- *speed*: `float`, rotating speed of the encoder motor EM1 (connected to the left wheel by default), ranging from `-200` to `+200` RPM

If you set this parameter to a positive value, the output shaft of EM1 rotates counterclockwise.

- *port*: `float` or `str`, port to which the encoder motor is connected

The default value is `"all"`. You can set this parameter to one of the following values:

`"all "``"em1 "``"em2 "``"ALL "``"EM1 "``"EM2 "``1``2`**MP**`mbot2.EM_turn(angle, speed, port)`

Sets the angle the encoder motor connected to the specified port turns counterclockwise

When this API is executed, the thread is blocked until the encoder motor turns the specified angle or the

`stop()` command is received.

Parameters:

- *angle*: `int`, number of degrees the specified encoder motor turns counterclockwise

- *speed*: `float`, rotating speed of the specified encoder motor when it turns counterclockwise, ranging from `0` to `+200` RPM
- *port*: `float` or `str`, port to which the encoder motor is connected
The default value is `"all"`. You can set this parameter to one of the following values:

`"all"``"em1"``"em2"``"ALL"``"EM1"``"EM2"``1``2`**MP**`mbot2.EM_get_angle(port)`

Obtains, in real time, the angle the encoder motor connected to the specified port turns counterclockwise

mBot2 resets the angles of the encoder motors when it starts. The number of degrees increases when an encoder motor turns counterclockwise and decreases when the encoder motor turns clockwise.

Parameter:

- *port*: `float` or `str`

You can set this parameter to one of the following values:

`"em1"``"em2"``"EM1"``"EM2"``1``2`

An `int` value ranging from `-∞` to `+∞` is returned, in degrees.

MP`mbot2.EM_get_speed(port)`

Obtains, in real time, the rotating speed of the encoder motor connected to the specified port when it turns counterclockwise

The rotating speed of the encoder motor may be lower than the set speed when the load is too heavy or may be higher than the set speed due to the action of external force.

Parameter:

- *port*: `float` or `str`

You can set this parameter to one of the following values:

`"em1"`

`"em2"`

`"EM1"`

`"EM2"`

`1`

`2`

An `int` value ranging from `-∞` to `+∞` is returned, in RPM.

MP

`mbot2.EM_get_power(port)`

Obtains, in real time, the power of the encoder motor connected to the specified port

Parameter:

- *port*: `float` or `str`

You can set this parameter to one of the following values:

`"em1"`

`"em2"`

`"EM1"`

`"EM2"`

`1`

`2`

An `int` value ranging from `-100` to `+100` is returned, in percentage.

MP

`mbot2.EM_reset_angle(port)`

Resets the angle of the encoder motor(s) connected to the specified port(s)

Parameter:

- *port*: `float` or `str`, port to which the encoder motor is connected

The default value is `"all"`. You can set this parameter to one of the following values:

`"all"`

`"em1 "``"em2 "``"ALL "``"EM1 "``"EM2 "``1``2``mbot2.EM_lock(is_lock = False, port)`

Enables the self-locking function of the encoder motor connected to the specified port
 When the self-locking function is enabled, the encoder motor attempts to stay in the current position after its motion. This function is disabled by default.

Parameters:

- *is_lock*: `bool`, whether to enable the self-locking function for the specified encoder motor

The default value is `"False"`.

- *port*: `float` or `str`, port to which the encoder motor is connected

The default value is `"all"`. You can set this parameter to one of the following values:

`"all"``"em1 "``"em2 "``"ALL "``"EM1 "``"EM2 "``1``2`

Driving DC motors

`mbot2.motor_add(power, port)`

Changes the power output of the motor connected to the specified port

Parameters:

- *power*: `float`, power to be changed for the motor, ranging from `-200` to `+20`

- *port*: `int` or `str`, port to which the motor is connected

You can set this parameter to one of the following values:

`"all"`

`"m1"`

`"m2"`

`"M1"`

`"M2"`

`1`

`2`

MP

`mbot2.motor_set(power, port)`

Sets the rotating speed of the motor connected to the specified port

Parameters:

- *power*: `float`, power of the motor; setting range: `-100~+100` %
When `power` > 0, the output shaft of the motor rotates counterclockwise; and when `power` < 0, the output shaft rotates clockwise. This may not be true for all the motors due to the features of motors. Some motors may require large start current and therefore can't work when `power` is low.
- *port*: `int` or `str`, port to which the motor is connected

Setting range:

`all`

`m1`

`m2`

`M1`

`M2`

`1`

`2`

MP

`mbot2.motor_get(port)`

Obtains the power of the motor connected to the specified port

Parameter:

- *port*: `float` or `str`, port to which the motor is connected

Setting range:

`m1`

`m2`

`M1`

M2

1

2

A `float` value ranging from `-100` to `100` is returned, in percentage.

MP

`mbot2.motor_drive(power1, power2)`

Sets the power output of the motors connected to ports M1 and M2

This API is used to set the power output of the motors connected to M1 and M2 and thus can synchronize the motion of the two motors.

Parameters:

- *power1*: `float`, power output of the motor connected to port M1; setting range: `-100` to `+100`, in percentage
- *power2*: `float`, power output of the motor connected to port M2; setting range: `-100` to `+100`, in percentage

MP

`mbot2.motor_stop(port)`

Sets the power output of the motor connected to the specified port to zero

Parameter:

- *port*: `float` or `str`, port to which the motor is connected

Setting range:

all

m1

m2

M1

M2

1

2

Driving servos

MP

`mbot2.servo_add(angle, port)`

Changes the angle of the servo connected to the specified port

If the API `mbot2.servo_set(angle, port)` has not been used before, when this API

is executed, the servo rotates to the position of 20 degrees first and then rotates the degrees set in this API.

Parameters:

- *angle*: `int`, number of degrees by which the angle of the servo is to be changed; setting range: `-180~+180`, in degrees
- *port*: `int` or `str`, port to which the servo is connected

Setting range:

`all`

`s1`

`s2`

`s3`

`s4`

`S1`

`S2`

`S3`

`S4`

`1`

`2`

`3`

`4`



`mbot2.servo_set(angle, port)`

Sets the angle of the servo connected to the specified port

Parameters:

- *angle*: `int`, number of degrees of the servo, ranging from `0~180`, in degrees
- *port*: `int` or `str`, port to which the servo is connected

Setting range:

`all`

`s1`

`s2`

`s3`

`s4`

`S1`

`S2`

`S3`

`S4`

`1`

`2`

`3`

4

MP

`mbot2.servo_get(port)`

Obtains the angle set for the servo connected to the specified port

The servo can't detect the angle itself, and therefore, if the API `mbot2.servo_release(port)` is executed or the servo is rotated manually, the angle obtained may not be the accurate one.

Parameter:

- *port*: `int` or `str`, port to which the servo is connected

Setting range:

`s1``s2``s3``s4``S1``S2``S3``S4``1``2``3``4`

An `int` value ranging from `0` to `180` is returned, in degrees.

MP

`mbot2.servo_release(port)`

Releases the angle of the servo connected to the specified port

When this API is executed, the output shaft of the servo is no longer locked until the API `mbot2.servo_add(angle, port)` or `mbot2.servo_set(angle, port)` is executed.

- *port*: `int` or `str`, port to which the servo is connected

Setting range:

`all``s1``s2``s3``s4`

S1

S2

S3

S4

1

2

3

4

MP`mbot2.servo_drive(angle1, angle2, angle3, angle4)`

Sets the angles of the servos connected to ports S1, S2, S3, and S4

This API is used to set the angles of the servos connected to S1, S2, S3, and S4, and thus can synchronize the motion of the servos.

Parameters:

- *angle1*: `int`, angle of the servo connected to port S1; setting range: `0-180`, in degrees
- *angle2*: `int`, angle of the servo connected to port S2; setting range: `0-180`, in degrees
- *angle3*: `int`, angle of the servo connected to port S3; setting range: `0-180`, in degrees
- *angle4*: `int`, angle of the servo connected to port S4; setting range: `0-180`, in degrees

Driving LED strips

MP`mbot2.led_on(r,g,b, id = "all", port)`

Lights up the LEDs on the LED strip or ring connected to the specified port in the specified color(s)

Parameters:

- *r*: `int` or `str`
 - r*: `int`, intensity of the red color; setting range: `0 - 255`
 - r*: `str`, full name or abbreviation of a color; the following describes colors and their abbreviations:

```

1  red r
2  orange o
3  yellow y
4  green g
5  cyan c
6  blue b
7  purple p
8  white w
9  black k

```

Python |  Copy

- *g*: `int` , intensity of the green color; setting range: `0-255`
- *b*: `int` , intensity of the blue color; setting range: `0-255`
- *id*: `int` or `str` ; the default value is `all`
 - id*: `str` , only the value `all` is valid
 - id*: `int` , setting range: `1-36` , indicating the position of the LED to be lit up
- *port*: `int` or `str` , port to which the LED strip or ring is connected

Setting range:

```

all
s1
s2
S1
S2
1
2

```

MP

`mbot2.led_show(color, port)`

Sets the color(s) of all LEDs on the LED strip or ring connected to the specified port

- *color*: `str` , color(s) of the LEDs, set in the "`color1 color2 color3 color4 color5...color36`" mode, with one space between any two adjacent colors. If you set more than 36 colors, only the first 36 colors are used. You can set this parameter to the full name or abbreviation of the colors. The options include the following:

```

red , r
green , g
blue , b
yellow , y
cyan , c
purple , p
white , w
orange , o

```

`black , k`

- *port*: `int` or `str` , port to which the LED strip or ring is connected

Setting range:

`all``s1``s2``S1``S2``1``2`

MP

`mboot2.led_move(step, cycle, port)`

Makes the colors of the LEDs on the LED strip connected to the specified port roll from left to right by the specified number of positions

Parameters:

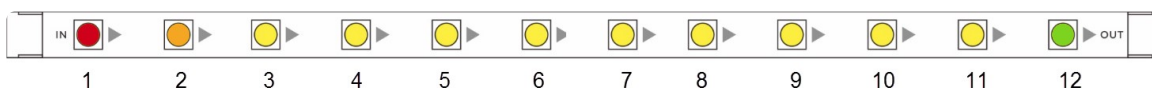
- *step*: `int` , number of positions by which the colors of the LEDs roll; setting range: `-36` to `+36` ; the default value is 1.
- *cycle*: `int` , range of the LED color rolling; setting range: `1-36`
- *port*: `int` or `str` , port to which the LED strip or ring is connected

Setting range:

`all``s1``s2``S1``S2``1``2`

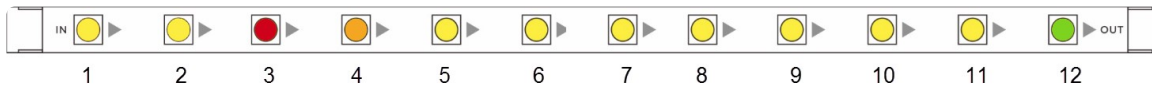
For example:

Set the initial colors of the LEDs on an LED strip (12 RGB LEDs) to "r o y y y y y y y y g", as shown in the following figure.



Then, set `step` to 2 and `cycle` to 4.

The LED strip is lit up as follows.



`mbot2.led_off(id = "all", port)`

Turns off the specified LED(s) on the LED strip or ring connected to the specified port

Parameters:

- *id*: `int` or `str`; the default value is `all`
 id: `int`, setting range: `1-36`, indicating the position of the LED to be turned off
 id: `str`, only the value `all` is valid
- *port*: `int` or `str`, port to which the LED strip or ring is connected

Setting range:

`all`

`s1`

`s2`

`S1`

`S2`

`1`

`2`



`mbot2.led_add_bri(brightness, port)`

Changes the brightness of the LED strip or ring connected to the specified port

Parameters:

- *brightness*: `int`, percentage by which the brightness of the LED strip or ring is to be changed; setting range: `-100` to `+100`, in percentage
- *port*: `int` or `str`, port to which the LED strip or ring is connected

Setting range:

`all`

`s1`

`s2`

`S1`

`S2`

`1`

`2`

**`mbot2.led_set_bri(brightness, port)`**

Sets the brightness of the LED strip or ring connected to the specified port

Parameters:

- *brightness*: `int`, the brightness of the LED strip or ring; setting range: `-100` to `+100`, in percentage
- *port*: `int` or `str`, port to which the LED strip or ring is connected

Setting range:

`all``s1``s2``S1``S2``1``2`**`mbot2.led_get_bri(port)`**

Obtains the brightness of the LED strip or ring connected to the specified port

Parameter:

- *port*: `int` or `str`, port to which the LED strip or ring is connected

Setting range:

`s1``s2``S1``S2``1``2`

Pin extension

**`mbot2.write_digital(val, port)`**

Sets the digital input for the specified pin(s)

Parameters:

- *val*: `int` or `bool`
- Setting range:

- `True/1` : high-level input
`False/0` : low-level input
- *port*: `int` or `str` , port where the pins are located

Setting range:

`all`

`s1`

`s2`

`S1`

`S2`

`1`

`2`



`mbot2.read_digital(port)`

Obtains the digital input at the specified pin

Parameter:

- *port*: `int` or `str` , port where the pins are located

Setting range:

`s1`

`s2`

`S1`

`S2`

`1`

`2`

An `int` value ranging from `0` to `1` is returned, where `0` indicates a low electrical level and `1` indicates a high electrical level.



`mbot2.set_pwm(duty, frequency, port)`

Sets the specified pin(s) to output PWM signals with the specified frequency and duty cycle

Parameters:

- *duty*: `int` , duty cycle of the PWM signals to be output; setting range: `0-100` , in percentage
- *frequency*: `int` , frequency of the PWM signals to be output; setting range: `1-200` `0` , in Hz
- *port*: `int` or `str` , port where the pins are located

Setting range:

`all`

2



Parameter:

- Setting range:

2

A `float` value ranging from `0` to `5` is returned, in volts.

%3A%20Extension%20parts%20are%20not%20included%20in%20the%20package%20of%20CyberPi.%2C%20you