EXP6 智能擴充板
上位機函式庫說明

對象：使用 SPIKE Prime 上課、比賽的老師與學生

適用：Pybricks 4.0 以上 · 官方 SPIKE App 3.4.3（教育版）· MINDSTORMS 10.x／SPIKE App 2.x（玩具版）

需搭配 EXP6 韌體 2026-11-03 以後 · 2026-11-09 · 少伯創意機器人有限公司（MBC Robot）

1. 開始之前

EXP6 讓 SPIKE Prime 的**一個孔**變成**六個孔**：可以接樂高馬達、樂高感應器，以及我們 MBC 的感應器（循行者、雷射、遙控器、神攝手）。

你需要準備

項目	說明
EXP6 擴充板	韌體要是 **2026-11-03 以後**的版本（舊韌體請先更新）
電池	外接 **2S 鋰電池**（7.4 V，充飽 8.4 V，XT30 接頭，需自備）。馬達的電來自這顆電池，不是 Hub。**不要接 3S 以上**：電池超過 9 V 時，EXP6 會停止供電給樂高顏色／超音波感應器來保護它們（感應器不亮、讀值沒有意義），馬達照常；函式庫會印一次警告。
函式庫檔案	Pybricks：`MBC_exp6_obj_Lib.py`；官方 App：先執行 `Install_MBC_exp6_Lib.py`

兩種孔號，別搞混

寫在哪裡	是哪一個孔	例子
建立擴充板物件時	**Hub** 的孔（EXP6 插在哪）	`MBC_EXP6(3)` = EXP6 插在 Hub 的 C 孔
其他所有函式	**EXP6** 的孔 1~6	`exp6.motor_run(2, 50)` = EXP6 的 2 號孔

> **Pybricks 的 Hub 孔號**
>
> 1 = A、2 = B、3 = C、4 = D、5 = E、6 = F。

2. Pybricks：快速開始

1. 打開 code.pybricks.com，把 `MBC_exp6_obj_Lib.py` 上傳到同一個專案。
2. 新增你的程式檔，第一行匯入函式庫。
3. 建立擴充板物件，選「一般模式」或「多工模式」。

一般模式 vs 多工模式

	一般模式	多工模式
建立物件	`MBC_EXP6(3, False)`	`MBC_EXP6(3, True)`
呼叫函式	直接呼叫	每個呼叫前加 `await`
等待	`exp6.keep_alive_wait(1000)`	`await wait(1000)`（背景自動維持連線）
適合	入門、一步一步做	同時做很多件事（邊走邊看感應器）

一般模式 SYNC

```
from MBC_exp6_obj_Lib import MBC_EXP6

exp6 = MBC_EXP6(3, False)       # EXP6 插在 Hub 的 C 孔

exp6.motor_run(1, 50)           # EXP6 1 號孔的馬達，速度 50
exp6.keep_alive_wait(2000)      # 等 2 秒（不要用 wait）
exp6.motor_stop(1, "brake")     # 煞車停下
```

多工模式 ASYNC

```
from pybricks.tools import wait, run_task
# 下一行要放在 pybricks.tools 之後
from MBC_exp6_obj_Lib import MBC_EXP6, run_task

exp6 = MBC_EXP6(3, True)

async def main():
    await exp6.motor_run(1, 50)
    await wait(2000)
    await exp6.motor_stop(1, "brake")

run_task(main())
```

為什麼不能用 wait()？

EXP6 有安全機制：超過 0.5 秒沒收到 Hub 的訊號，馬達會自動停下（避免程式當掉時機器人亂跑）。一般模式請用 `keep_alive_wait(ms)` 取代 `wait(ms)`；長時間計算中間呼叫一次 `keep_alive()`。多工模式由背景自動處理。

3. 馬達控制

速度、功率都是 **-100～100**，正數順時針、負數逆時針。

函式	做什麼	參數
`motor_run(port, speed)`	定速轉動（自動維持速度，推薦）	speed -100~100
`motor_power(port, power)`	直接給功率（不管速度）	power -100~100
`motor_stop(port, stop)`	停止	stop 見下表
`motor_run_degrees(port, speed, degrees, stop)`	轉指定角度	degrees -2499~2499
`motor_track_target(port, speed, angle, stop)`	轉到馬達的絕對朝向（走最短路徑）	angle 0~359
`drive(left, right, left_speed, right_speed)`	兩顆馬達同步（車子走直線不偏）	兩個孔不能相同
`drive_stop(left, right, stop)`	兩顆一起停	stop 見下表
`stop_all(stop)`	六個孔的馬達全部停	stop 見下表
`set_motor_inverted(port, True)`	馬達反裝（之後正負自動相反）	True 反裝／False 正裝
`motor_set_pid(port, p, i, d)`	調整控制參數（進階）	0~100，**50 = 原廠值**

stop：四種停法

寫法	名稱	效果
`1` 或 `"coast"`	滑行	放開，馬達慢慢停
`2` 或 `"brake"`	煞車	很快停下，但用手還轉得動
`3` 或 `"hold"`	鎖死	停在原地並出力抵抗
`4` 或 `"continue"`	續轉／維持現狀	轉角度：到位後繼續以原速轉；停止函式：什麼都不做

轉角度的方向怎麼決定

`motor_run_degrees` 的方向 = **速度的正負 × 角度的正負**：同號順時針、異號逆時針。例如 `(60, -360)` 是逆時針轉一圈。

範例：轉一圈後鎖死，再轉到朝向 90 度 SYNC

```
exp6.motor_run_degrees(1, 60, 360, "hold")
exp6.keep_alive_wait(1500)
exp6.motor_track_target(1, 40, 90, "hold")
exp6.keep_alive_wait(1000)
```

> **轉角度不會等到轉完**
>
> 指令送出就往下執行。要等轉完，用 `keep_alive_wait()`，或用 `get_motor_angle()` 檢查角度。

範例：車子前進 2 秒、原地右轉 ASYNC

```
async def main():
    await exp6.drive(1, 2, 50, 50)     # 左輪 1 號孔、右輪 2 號孔
    await wait(2000)
    await exp6.drive(1, 2, 40, -40)    # 原地右轉
    await wait(600)
    await exp6.drive_stop(1, 2, "brake")
```

4. 樂高感應器與馬達讀值

函式	回傳
`get_motor_angle(port)`	馬達累加角度（度），可用 `reset_angle(port)` 歸零
`get_motor_abs_angle(port)`	馬達絕對朝向 0~359（不受歸零影響）
`get_motor_speed(port)`	馬達目前速度（度／秒）
`get_color_color(port)`	顏色代碼（0~10）
`get_color_reflection(port)`	反射光 0~100（循線用）
`get_color_rgb(port)` / `get_color_hsv(port)`	(R, G, B) 各 0~100 / (H 0~359, S, V)
`get_color_red/green/blue(port)`、`get_color_hue/sat/val(port)`	單一通道
`get_ultrasonic_distance(port)`	超音波距離（mm）
`get_touch_force(port)`	觸碰感應器按壓力量 0~100
`get_device_id(port)`	這一孔接什麼：0 沒有、1 觸碰、3 顏色、4 超音波、5 馬達、6 MBC 感應器
`get_voltage()`	擴充板電池電壓（V）

範例：看到黑線就停 SYNC

```
exp6.drive(1, 2, 30, 30)
while exp6.get_color_reflection(3) > 20:    # 顏色感應器在 3 號孔
    exp6.keep_alive()
exp6.drive_stop(1, 2, "brake")
```

5. MBC 感應器

MBC 感應器插在 EXP6 上，函式名稱跟它**直接插 Hub 時一樣**。差別只在建立方式：用 EXP6 物件「宣告」它插在哪一孔。

1. 感應器**直接插上就能用**（跟插在主機上一樣，不用另外設定）。
2. 插到 EXP6 的任一孔。
3. 在程式開頭（多工模式要在 `run_task` 之前）宣告：

宣告 MBC 感應器 SYNC

```
exp6  = MBC_EXP6(3, False)
line8 = exp6.line8(2)        # 循行者 8 路插在 EXP6 的 2 號孔
tof8  = exp6.laser8(4)       # 八方位雷射插在 4 號孔

print(line8.pos100(), tof8.min_distance())
```

宣告寫法	感應器
`exp6.line8(port)`	循行者 8 路
`exp6.line16(port)`	循行者 16 路
`exp6.laser2(port)`	兩路雷射 TOF2
`exp6.laser8(port)`	八方位雷射 TOF8
`exp6.ps2(port)` / `exp6.ps4(port)`	PS2 遙控接收器／PS4 藍牙遙控
`exp6.unicam(port, mode)`	神攝手通用版，mode 見 5.6

讓程式跑得快的小技巧

每個函式都只讀感應器的「一小組資料」。同一組的函式（下表同一格底色）連續呼叫最快；換到另一組，第一次讀會多約 20～30 ms（官方 App 約 40～100 ms）。例如循線時一直讀 `pos100()` 很快；`pos100()` 和 `junction()` 是同一組，一起讀也很快。

讀不到時

感應器沒插好時，函式會回傳 `None`，並在主控台印一行提示。

5.1 循行者 8 路 `line8`

函式	回傳	同一組
`pos8()`	線的位置 -8～+8（0 = 正中間）	A
`width()`	壓到線的感應器數 0～8	
`pos100()`	高解析位置 -100～+100	B
`junction()`	0 沒線、1 單線、2 以上 = 分岔／路口	
`junction_name()`	`"none"` / `"line"` / `"junction"`	
`bin_raw()`	8 路黑白狀態 0～255（bit0 = 第 1 路）	C
`on_sensors()`	壓到線的通道清單，例如 `[4, 5]`	
`ir_ch(ch)`	第 ch 路光值 0～100（ch = 1～8）	每 2 路一組

範例：用循行者 8 路循線 SYNC

```
exp6.set_motor_inverted(2, True)  # 右輪跟左輪面對面裝，設成反裝，之後正數都是「前進」
line8 = exp6.line8(3)             # 循行者 8 路插在 3 號孔
while True:
    p = line8.pos100()            # 線在左邊是正數、在右邊是負數
    if p is None:                 # 讀不到就先停
        exp6.drive_stop(1, 2, "brake")
        continue
    turn = int(p * 0.3)           # 線偏左 → 左輪慢、右輪快
    exp6.motor_run(1, 40 - turn)
    exp6.motor_run(2, 40 + turn)
```

5.2 循行者 16 路 `line16`

函式	回傳	同一組
`pos16()` / `width()`	位置 -16～+16 / 壓線數 0～16	A
`pos100()` / `junction()` / `junction_name()`	同 8 路	B
`bin_raw()` / `on_sensors()`	16 路黑白 0～65535 / 通道清單	C
`ir_ch(ch)`	第 ch 路光值 **0～15**（ch = 1～16）	每 4 路一組

5.3 兩路雷射 `laser2`

函式	回傳	同一組
`left()` / `right()` / `ch(ch)`	距離 0～120 cm（120 = 前面沒東西）	A（全部同一組）
`average()` / `min_distance()`	兩邊平均／比較近的那邊	
`min_side()`	-1 左邊近、+1 右邊近、0 一樣	
`diff()`	左 − 右	
`is_ok(ch)`	這一路雷射正常嗎（True/False）	B

5.4 八方位雷射 laser8

函式	回傳	同一組
`ch(ch)`	第 ch 方位距離 0～120 cm（ch = 1～8）	每 2 方位一組
`min_direction()` / `min_distance()`	最近障礙物的方向 -8～+8（0 = 正前方）／距離	A
`max_direction()` / `max_distance()`	最空曠的方向／距離	B
`init_status()` / `fault_status()` / `is_ok(ch)`	各方位開機是否成功／目前是否故障	C

5.5 PS2／PS4 遙控 ps2 / ps4

函式	回傳	同一組
`pressed(name)`	這顆鍵有沒有按（True/False）	A
`pressed_buttons()`	正在按的鍵名清單	
`buttons()`	16 個按鍵的數字（進階）	
`lx()` / `ly()`	左搖桿 1～9（5 = 中間，往右／往上變大）	B
`rx()` / `ry()`	右搖桿 1～9	C

按鍵名稱：`"up"` `"down"` `"left"` `"right"`、`"triangle"` `"circle"` `"cross"` `"square"`、`"l1"` `"l2"` `"r1"` `"r2"` `"l3"` `"r3"`、`"start"` `"select"`。

範例：搖桿開車，按 ✕ 結束 SYNC

```
ps = exp6.ps4(4)
while not ps.pressed("cross"):
    y = ps.ly()                      # 1~9
    if y is None:
        continue
    speed = (y - 5) * 20             # -80 ~ +80
    exp6.drive(1, 2, speed, speed)
exp6.drive_stop(1, 2, "brake")
```

5.6 神攝手 unicam

寫法跟神攝手直插 Hub 的 `UniCam(port, mode)` 一樣：`cam = exp6.unicam(port, mode)`。**mode 一定要跟相機選單切的一樣**（主機讀不到相機目前在哪個模式）；mode＝相機模式 ×10＋子選項：

mode	相機模式・子選項
11 / 12 / 13 / 14	顏色追蹤：最多色（14 色）／多色（7 色）／標準 XY＋面積（4 色）／精細 XYWH（2 色）
21 / 22 / 23	座標取色：色相（14 點）／色相＋飽和（7 點）／色相＋飽和＋明度（4 點）
30	照順序給色號（由左到右，最多 13 塊）
41 / 42 / 43	循線：黑線／白線／色線
51 / 52	物體偵測：物體清單（4 個）／物體詳細（2 個）

座標是像素 X 0～159（左→右）、Y 0～119（上→下），(0, 0)＝沒找到；色相碼 0～255（0～240 色相、241～248 白／灰、249～255 黑）。用錯模式的函式會直接報錯，告訴你要用哪個模式。

函式	回傳	模式	讀幾次
`xy(c)` `x(c)` `y(c)` `found(c)`	第 c 色的位置；模式 11 是 16×12 格的格子中心（±5 像素）	11～14	1
`cell(c)`	16×12 格 (1～16, 1～12)	11	1
`area(c)`	面積（佔畫面 %）	13、14	1
`size(c)`	(寬, 高) 像素	14	1
`count(c)` `colors_found()`	第 c 色有幾塊／16 色裡找到幾色（模式 11、12 要換到感應器的另一個 mode，第一次多約 30 ms）	11～14	1
`max_color()` `max_point()` `max_object()`	這個模式給幾色／幾點／幾個物體（不讀感應器）	各自的模式	0
`hue(i)` `sat(i)` `val(i)` `color(i)`	第 i 個取樣點的色相碼、飽和度、明度（0～100）、顏色名稱	21～23	1
`hsv(i)`	(色相碼, 飽和度, 明度)，這個模式沒給的是 `None`	21～23	1～2
`order()` / `total()`	由左到右每塊的色相碼（list）／總塊數	30	7／1
`near()` `mid()` `far()` `angle()`	近／中／遠偏移 −100～100（線在左＝負）、線角度 −90～90	41～43	1
`width()` `flags()` `junction()` `junction_dist()`	近處線寬 %、有線旗標（近 1＋中 2＋遠 4）、路口類型（0 無 1 十字 2 T 3 左 4 右 5 盡頭）、路口距離 %	41～43	1
`edges()` `far_width()` `marker()` `lost()`	(左緣, 右緣)、遠處線寬 %、(標記線色號, 條數)、最後離線方向（0 有線 1 左 2 右）	41～43	1
`object(i)` `object_found(i)` `object_color(i)` `object_shape(i)`	物體代碼（色號×10＋形狀）、有沒有、色號 1～8、形狀 0～9（`shape_name()` 換名稱）；物體 1＝面積最大	51、52	1
`object_xy(i)` / `object_count()`	質心 (X, Y) 像素／畫面裡有幾個物體	51、52／	1

		51	
`object_size(i)` `object_area(i)`	外框 (寬, 高) 像素、面積 %	52	1
`sensor_ok()` `is_unicam()` `firmware_version()` `device_id()`	讀得到嗎、是不是神攝手、韌體版本 (主, 次)、(0x4C, 3, 韌體版字)	全部	1～2

範例：追色 1，色塊在左就左轉 SYNC

```
cam = exp6.unicam(5, 13)           # 相機選單切到 1-3（顏色追蹤 標準）
while True:
    x, y = cam.xy(1) or (0, 0)     # 讀不到時回傳 None
    if x == 0:
        exp6.drive(1, 2, 30, -30)   # 沒看到：原地轉找
    else:
        turn = (x - 80) // 2        # 80 = 畫面中間
        exp6.drive(1, 2, 40 + turn, 40 - turn)
```

EXP6 上的神攝手跟直插 Hub 不同的地方

①讀不到時回傳 `None`（直插版回傳上一次的值）②沒有 `line_all()`，循線的每個值都有自己的函式 ③`order()` 要讀整組 14 bytes（約 120 ms，官方 App 約 280 ms），只要總塊數用 `total()` 比較快。

只支援神攝手通用版；舊的神攝手 WRO 版（HUE12）、學校版不支援插 EXP6。

直插 Hub 才有、EXP6 上沒有的函式

一次回傳整組資料的函式：`read_all()`、`ir_calibration()`、`distances()`、神攝手的 `line_all()`。要整組資料時，用迴圈逐一讀 `ir_ch(ch)`、`ch(ch)`。

目前不支援：IMU 陀螺儀、羅盤。

多工模式的寫法 ASYNC

```
exp6  = MBC_EXP6(3, True)
line8 = exp6.line8(2)            # 宣告要在 run_task 之前

async def main():
    while True:
        p = await line8.pos100()
        print(p)
        await wait(50)

run_task(main())
```

5.7 對 MBC 感應器下命令 cmd

插在 EXP6 上的 MBC 感應器也能下命令（例如改循行者的門檻、左右反向），命令碼和參數跟感應器**直接插 Hub 時一樣**（見各感應器的說明）。Pybricks 版、官方 App 版的函式名稱相同。

函式	做什麼
`cmd(code, p1, …)`	送出命令。`code` = 命令碼 0xE0～0xEE，參數最多 5 個（-9999～9999）。回傳這次的序號。同命令、同參數不會重送；要重做加 `again=True`
`wait_done(timeout=10000)`	等命令做完：`True` 完成；`False` 失敗、拒絕或等超過 timeout 毫秒
`cmd_state()`	0 完成、1 執行中、2 失敗（參數不對）、3 拒絕（這款沒有這個命令）；感應器還沒收到 = `None`
`busy()`	最後一個命令還沒做完 = `True`

把循行者的二值化門檻改成 60 SYNC

```
line8 = exp6.line8(2)
line8.cmd(0xE4, 60)              # E4 = 二值化門檻（1～99，開機回到 50）
if line8.wait_done():
    print("改好了")
else:
    print("沒成功，狀態", line8.cmd_state())
```

會寫進感應器的命令

循行者的黑白校準 `0xE2` ／ `0xE3`、5 秒校準 `0xE7` 會存進感應器，練習時不要亂送。門檻 `0xE4`、左右反向 `0xE9` 開機就回到預設。

速度

實測一個命令從送出到收到結果約 0.1～0.2 秒。多工模式（Pybricks）要 `await line8.cmd(…)`、`await line8.wait_done()`。

6. 官方 SPIKE App（教育版 3.4.3）

安裝（每台 Hub 做一次）

1. 在 SPIKE App 開一個 Python 專案，貼上 `Install_MBC_exp6_Lib.py` 的內容。
2. 執行，看到「安裝成功」就完成。函式庫會存在 Hub 裡，之後每個專案都能用。
3. 不要了就執行 `Uninstall_MBC_exp6_Lib.py`。

官方 App 範例 SPIKE APP

```
from hub import port
import time
from MBC_exp6_SPIKE_App_Lib import MBC_EXP6

exp = MBC_EXP6(port.F)          # EXP6 插在 Hub 的 F 孔
time.sleep(1)

exp.motor_run(1, 50)
time.sleep(2)
exp.motor_stop(1, "brake")

line8 = exp.line8(2)            # MBC 感應器的用法跟 Pybricks 一樣
print(line8.pos100())
```

跟 Pybricks 版的差別

項目	官方 App 版
建立物件	`MBC_EXP6(port.F)`，用 `hub.port` 的孔
多工模式	沒有，不用 `await`；等待用 `time.sleep()`
顏色感應器	只拿得到 RGB，`get_color_reflection()` 是亮度近似值，**循線門檻要重調**
`get_device_id()`	跟 Pybricks 版一樣：0 無、1 力量、3 顏色、4 超音波、5 馬達、6 MBC 感應器
MBC 感應器	函式名稱相同；換到另一組資料第一次約多 40～100 ms
對 MBC 感應器下命令	函式名稱相同（`cmd`、`wait_done`…，見 5.7）

官方 App 範例共 10 支（`官方APP\通用範例\`）：01 馬達基礎、02 馬達進階、03 雙馬達底盤、04 距離／力量、05 顏色、06 系統、07 正反裝、08 停止方式、09 MBC 感應器、10 對 MBC 感應器下命令。

7. MINDSTORMS Robot Inventor（玩具版 51515）

適用 **LEGO MINDSTORMS Robot Inventor App 10.x** 的 Python（2026-11-05 上機驗證）。函式名稱、用法跟教育版**完全一樣**，只有安裝方式和匯入那一行不同。Robot Inventor 的積木不能用這個函式庫。

安裝（每台 Hub 做一次；韌體更新後要再裝）

1. 主機**用 USB 線**接電腦（用藍牙傳安裝程式時主機可能會關機），在 App 連上主機。
2. 開一個 Python 專案，貼上**玩具版資料夾**裡的 `Install_MBC_exp6_Lib.py`，執行，看到「安裝成功」就完成。函式庫存在主機的 `/projects/mbc_exp6/`。
3. 不要了就執行玩具版的 `Uninstall_MBC_exp6_Lib.py`。

玩具版程式開頭 ROBOT INVENTOR

```
from hub import port
import time
from projects.mbc_exp6.MBC_exp6_SPIKE_App_Lib import MBC_EXP6   # ← 只有這行跟教育版不同

exp = MBC_EXP6(port.F)          # EXP6 插在 Hub 的 F 孔
exp.motor_run(1, 50)
time.sleep(2)
exp.motor_stop(1, "brake")
```

跟教育版不一樣的 3 件事

項目	玩具版
EXP6 的通訊模式	程式第一次建立 `MBC_EXP6` 後，EXP6 會一直用玩具版的模式，**直到拔掉 EXP6 電池**。這段時間 App 裝置頁看不到 EXP6；換回教育版主機前先拔電池重開
心跳	函式庫在背景自動處理，不用寫
顏色感應器	讀得到感應器自己的反射值和顏色碼（比教育版的 RGB 近似值準），循線門檻跟 Pybricks 版一樣
程式結束後	馬達還在轉的話，約 10 秒內自動停。要讓馬達轉超過 10 秒又不呼叫函式，用 `exp.keep_alive_wait(毫秒)`

> **範例**
>
> 玩具版資料夾的 `範例\` 有 10 支，跟教育版同內容，只改了匯入那一行。第一次用某一類 MBC 感應器時，主機會先載入那部分的函式庫（約 1 秒）。

8. 常見問題

馬達轉一下就自己停了

一般模式用了 `wait()`。改成 `keep_alive_wait()`；長迴圈裡呼叫 `keep_alive()`。

印出「未偵測到擴充板」

確認 EXP6 有接電池、建立物件時的 Hub 孔號對不對。插上後函式庫會自動重新連線。

MBC 感應器一直回傳 None

① 感應器有沒有插好 ② 宣告的孔號是不是 EXP6 的孔（1～6）③ EXP6 韌體是不是 2026-11-03 以後的版本 ④ 宣告的型號對不對（例如把 8 路宣告成 `line16` 會讀到亂碼）。

讀 MBC 感應器有點慢

一直換讀不同組的函式就會慢。同一組的函式放在一起讀；只需要位置就只讀 `pos100()`。

拔掉感應器再插回去

不用重跑程式，下一次讀值會自動恢復。

車子走直線會偏

用 `drive()` 而不是兩個 `motor_run()`；一邊馬達裝反時用 `set_motor_inverted()`，不要自己把速度改負的。

9. 函式速查表

分類	函式
連線	`MBC_EXP6(hub_port, multitask)` `keep_alive_wait(ms)` `keep_alive()` `run_task(main())`
馬達	`motor_run` `motor_power` `motor_stop` `motor_run_degrees` `motor_track_target` `drive` `drive_stop` `stop_all` `set_motor_inverted` `motor_set_pid` `reset_angle`
馬達讀值	`get_motor_angle` `get_motor_abs_angle` `get_motor_speed`
樂高感應器	`get_color_color` `get_color_reflection` `get_color_rgb` `get_color_hsv` `get_color_red/green/blue` `get_color_hue/sat/val` `get_ultrasonic_distance` `get_touch_force`
系統	`get_device_id` `get_voltage` `get_port_raw`
MBC 感應器	`line8` `line16` `laser2` `laser8` `ps2` `ps4` `unicam`（各自的函式見第 5 章）
對 MBC 感應器下命令	`cmd(code, …)` `wait_done()` `cmd_state()` `busy()`（見 5.7）

範例程式：Pybricks `pyBricks/通用範例/01～11`（11 = MBC 感應器）；官方 App `官方APP/通用範例/01～10`（09 = MBC 感應器、10 = 下命令）；玩具版 `官方APP/玩具版/範例/01～10`（只差匯入那一行）。