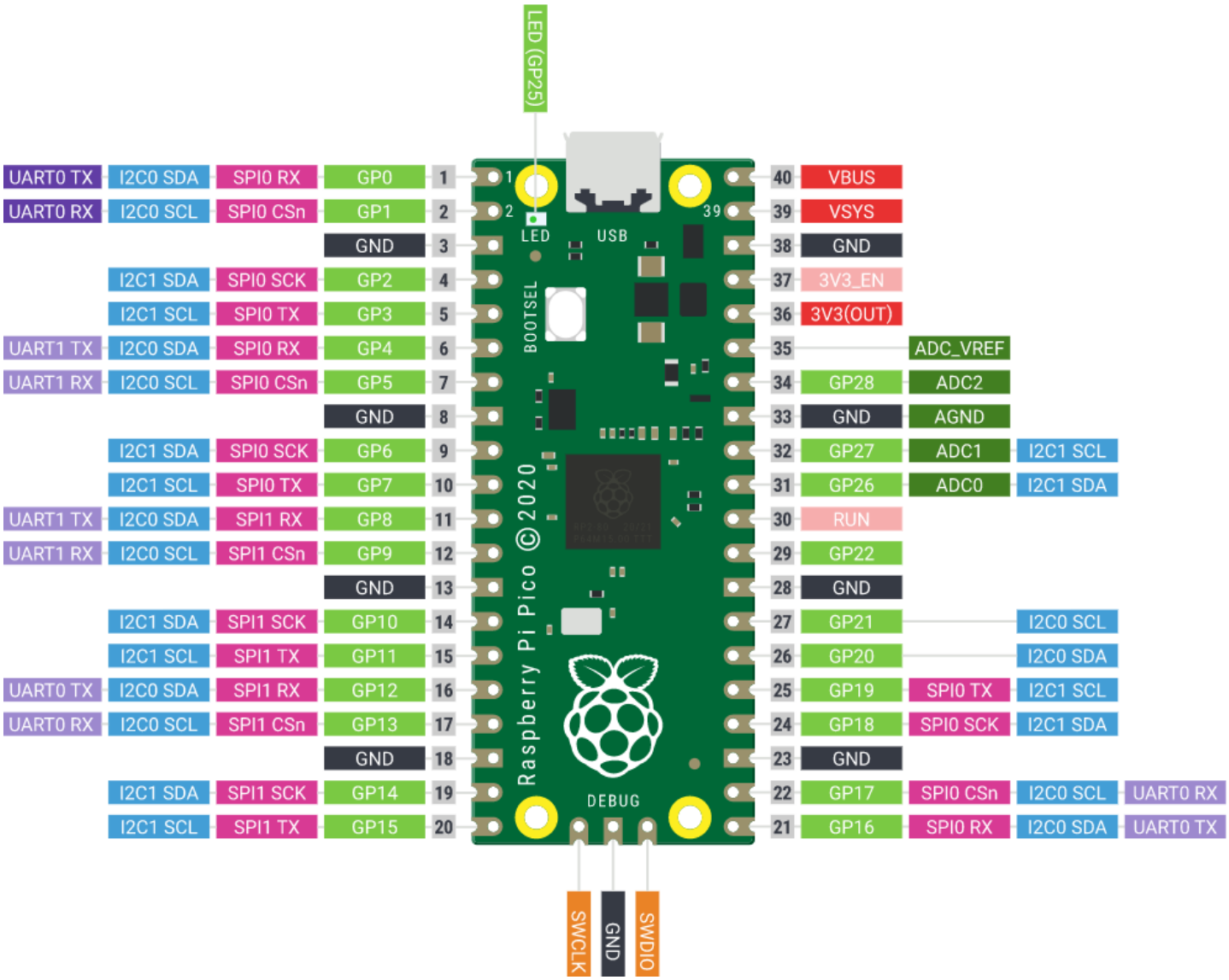
Raspberry Pi Pico 是一款具備靈活腳位，且低成本、高性能的開發板，價格僅 4 元美金，商品特色如下：

1. 採用 Raspberry Pi 英國設計的 RP2040 微控制器，雙核 Arm Cortex M0 +處理器，運行頻率 133 MHz
2. 264KB 的 SRAM 和 2MB 的片上 Flash
3. 支持低功耗睡眠和休眠模式
4. 能通過 USB 使用大容量儲存進行拖放式下載程式
5. 多達 26 個多功能 GPIO 引腳
6. 2 個 SPI，2 個 I2C，2 個 UART，3 個 12 位元的 ADC，16 個可程式控制的 PWM
7. 精準的時鐘和計時器與內建溫度感測器
8. 8 個可透過程式撰寫 I / O（PIO）狀態機，支持自定義外設備
9. 支援 C / C ++ 和 MicroPython 開發
10. 可執行 TensorFlow Lite 框架

外觀與腳位定義如下：(※若元件需使用 5V 電壓，則使用 Pin40 的 VBUS)



- VBUS - 這是來自 microUSB 匯流排的電源，5 V。如果 Pico 不是由 microUSB 聯結器供電，那麼這裡將沒有輸出。
- VSYS - 這是輸入電壓，範圍為 2 至 5 V。板載電壓轉換器將為 Pico 將其改為 3.3 V。
- 3V3 - Pico 內部調節器的 3.3 伏輸出。只要將負載保持在 300mA 以下，它就可用於為其他元件供電。
- 3V3_EN - 你可以使用此輸入禁用 Pico 的內部電壓調節器，從而關閉 Pico 和由其供電的任何元件。
- RUN - 可以啟用或禁用 RP2040 微控制器，也可以將其復位。

Pico 的 BOOTSEL 模式位於 RP2040 晶片內部的唯讀存儲槽中，不會被意外覆蓋。任何情況下按住 BOOTSEL 按鈕並插入 Pico 時，都會以驅動器的模式出現，可以在其中拖動新的 UF2 韌體文件，但無法藉由軟體編寫程式。不過在某些情況下可能需要確保淨空閃存，您可以藉由大容量存儲模式將特殊的 UF2 二進製文件拖放到您的 Pico 上格式化閃存。

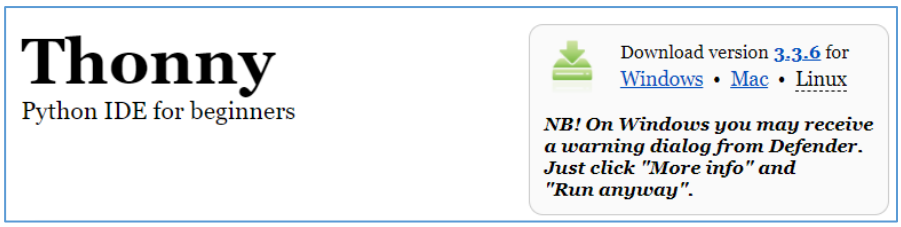
※ 硬體基本測試，不可以撰寫程式控制，步驟如下：

1. 下載 blink.uf2 韌體檔案文件。
2. 按住 BOOTSEL 按鈕，將 Pico 插入電腦的 USB 埠，連接 Pico 後，鬆開 BOOTSEL 按鈕。
3. 連接後會出現名為 RPI-RP2 的大容量存儲設備。
4. 將 blink.uf2 檔案文件拖曳進 RPI-RP2 內，Pico 會重新啟動，內建 GPIO25 開始閃爍。現在，MicroPython 會開始運作。

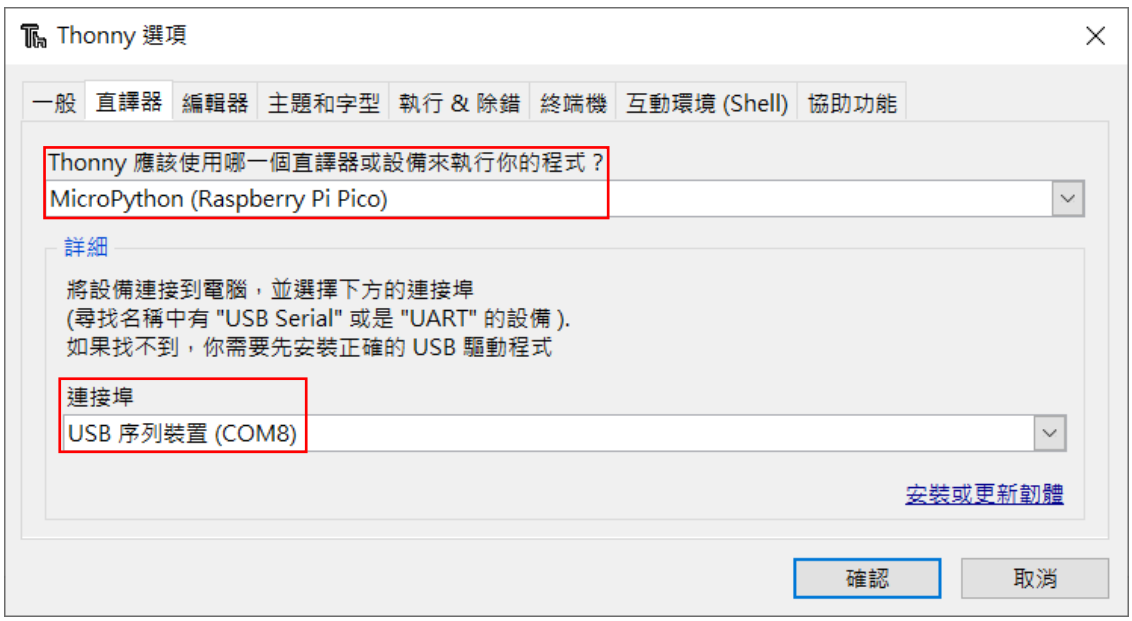
※ 建立 MicroPython 程式控制環境，步驟如下：

1. 下載 rp2-pico-20210324-unstable-v1.14-121-g4fc2866f4.uf2 韌體檔案文件。
2. 按住 BOOTSEL 按鈕，將 Pico 插入電腦的 USB 埠，連接 Pico 後，鬆開 BOOTSEL 按鈕。
3. 連接後會出現名為 RPI-RP2 的大容量存儲設備。
4. 將 rp2-pico-20210324-unstable-v1.14-121-g4fc2866f4.uf2 檔案文件拖曳進 RPI-RP2 內，Pico 會重新啟動，MicroPython 才能開始運作。
5. 本機右鍵→內容→裝置管理員，查看 COM?位置，若還看到應是驅動程式有誤，請下載 Pico_devices_cdc.inf，更新驅動程式後即可看到。

6. 到 https://thonny.org/ 下載編輯軟體，至少要為 3.3.3 版本以上才有支援 pico。



7. 進入 Thonny 主程式，『執行→選擇直譯器』，選擇"MicroPython(Raspberry Pi Pico)"直譯器與 COM?，最後按確定。



數位輸出測試

※ A01_內建 LED 閃爍.py 程式碼如下：

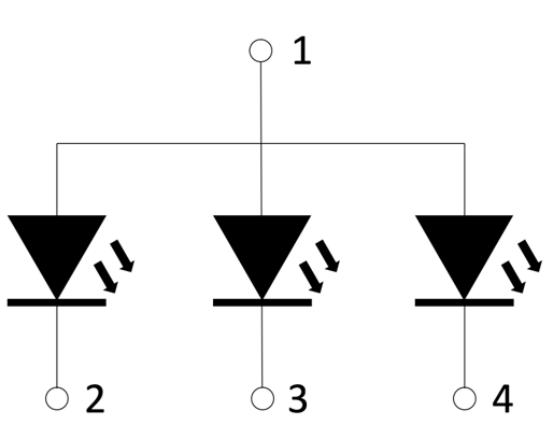
```
from machine import Pin
import utime
LED = Pin(25, Pin.OUT)
while True:
    LED.value(1)
    utime.sleep(0.5)
    LED.value(0)
    utime.sleep(0.5)
```

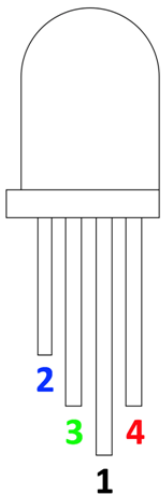
註 1：import machine 是用來設定 Pi Pico 所有相關硬體參數，若板子沒有正確連線或者直譯器沒有選對，則執行程式後會出現 ”import machine module named not found” 的錯誤。

註 2：import utime 目的是導入時間相關類別，因為後面 utime.sleep(0.5) 才能正常使用。

※ A02_RGB_LED.py 程式碼如下：(註：數位輸出僅能隨機顯示 8-1 種顏色，因為黑色代表不亮)

```
from machine import Pin
import utime, random
Red = Pin(16, machine.Pin.OUT)
Green = Pin(17, Pin.OUT)
Blue = Pin(18, Pin.OUT)
while True:
    r = random.randint(0, 1)
    g = random.randint(0, 1)
    b = random.randint(0, 1)
    Red.value(r)
    utime.sleep(0.1)
    Green.value(g)
    utime.sleep(0.1)
    Blue.value(b)
    utime.sleep(0.1)
```





類比輸出測試

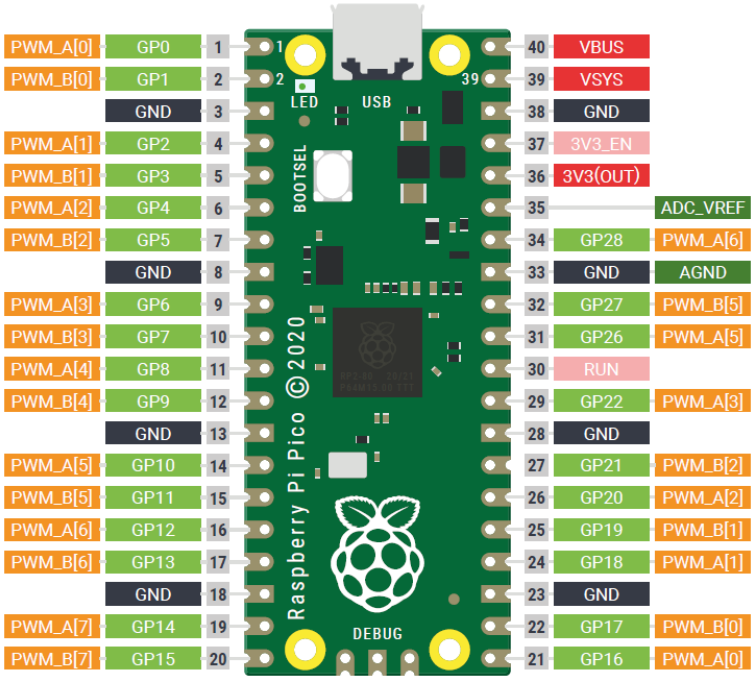
※ A03_內建 LED 漸亮.py 程式碼如下：

```
from machine import Pin, PWM
import utime
ch = PWM(Pin(25))                # PWM at GP25
ch.freq(1000)                    # Frequency = 1000Hz
i = 0
while True:
    ch.duty_u16(i)               # Change duty cycle
    utime.sleep_ms(300)          # Delay 300ms
    i = i + 5000                 # Increment i
    if i > 65535:
        i = 0
```

※ A04_RGB_LED 全彩.py 程式碼如下：(註：RGB 分別接於 GP16.17.18)

```
import time
from machine import Pin, PWM
pwm_pins = [16, 17, 18]   # set PWM pins
pwms = [PWM(Pin(pwm_pins[0])), PWM(Pin(pwm_pins[1])), PWM(Pin(pwm_pins[2]))]   # pwm array
[pwm.freq(1000) for pwm in pwms]        # set pwm freqs
step_val = 64    # step value for 16-bit breathing
range_0 = [ii for ii in range(0, 2**16, step_val)]          # brightening
range_1 = [ii for ii in range(2**16, -step_val, -step_val)]  # dimming
while True:
    for pwm in pwms:
            for i in range_0 + range_1:
                pwm.duty_u16(i)         # set duty cycle out of 16-bits
                time.sleep(0.001)       # sleep 1ms between pwm change
    for i in range_0 + range_1:
        for pwm in pwms:
            pwm.duty_u16(i) # set duty cycle
        time.sleep(0.001)       # wait 1ms
```

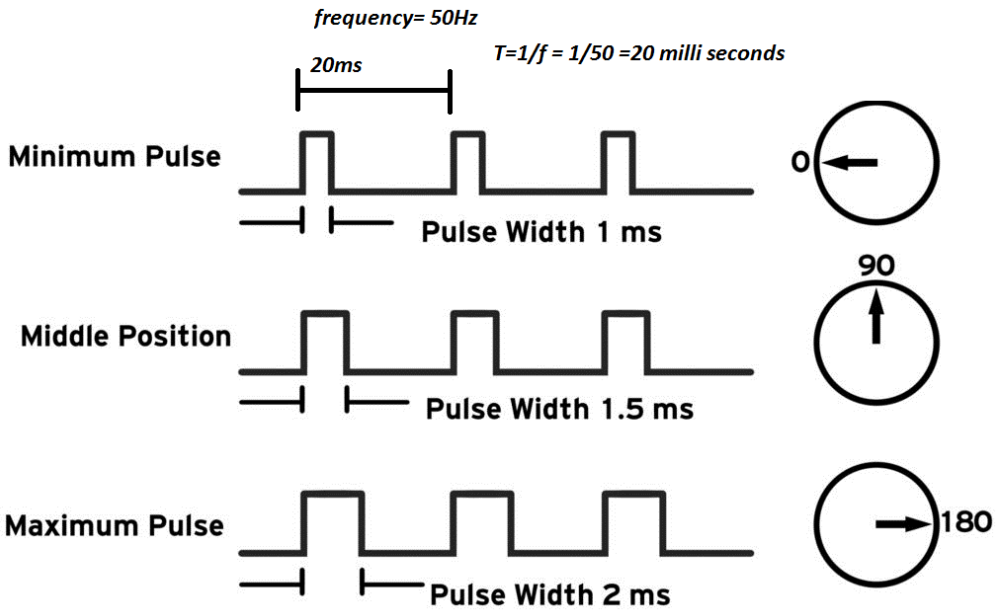
註：pico 的 PWM 腳位僅有 16 個，如右圖，所以有些腳位是共用，如 GP0 與 GP16 都是 PWM_A[0]，所以不可同時使用。



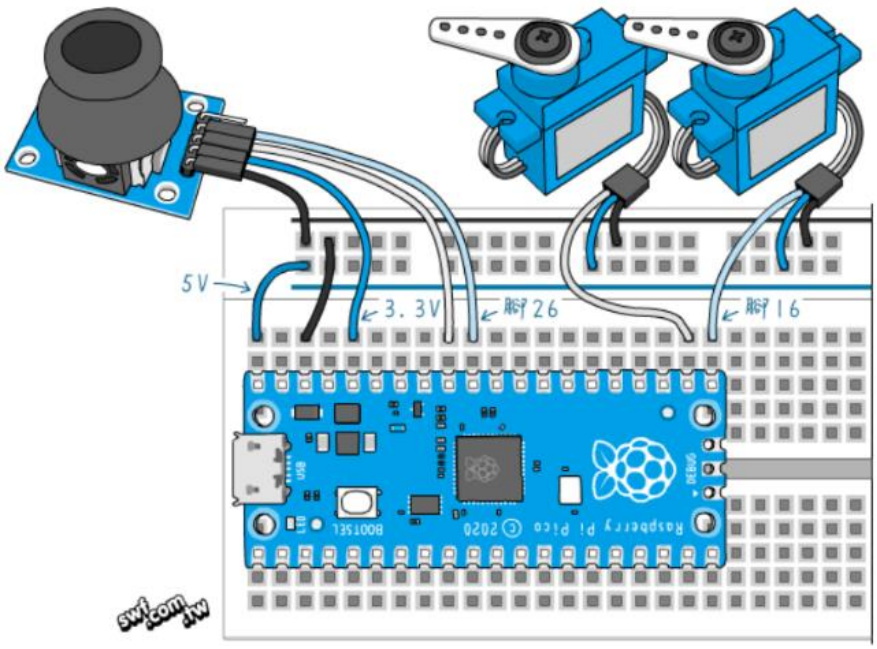
※ A05_不連續 SERVO 伺服馬達左右搖擺.py 程式碼如下：

註：不連續 SERVO 伺服馬達 0 度的 duty cycle=1700，180 度的 duty cycle=8000，90 度為中間值 4850，其餘角度請按比例計算。

```
from machine import Pin, PWM
import time
MIN_DUTY = 1700
MAX_DUTY = 8000
pwm = PWM(Pin(9))
pwm.freq(50)
duty = MIN_DUTY
direction = 1
pwm.duty_u16(1700)
time.sleep(1)
pwm.duty_u16(4850)
time.sleep(1)
pwm.duty_u16(8000)
time.sleep(1)
while True:
    for _ in range(1024):
        duty += direction
        if duty > MAX_DUTY:
            duty = MAX_DUTY
            direction = -direction
        elif duty < MIN_DUTY:
            duty = MIN_DUTY
            direction = -direction
        pwm.duty_u16(duty)
        time.sleep(0.001)
```



▼ 結合類比輸入，動動腦完成底下功能



數位輸入測試

※ A06_按鈕開關控制內建 LED.py 程式碼如下：

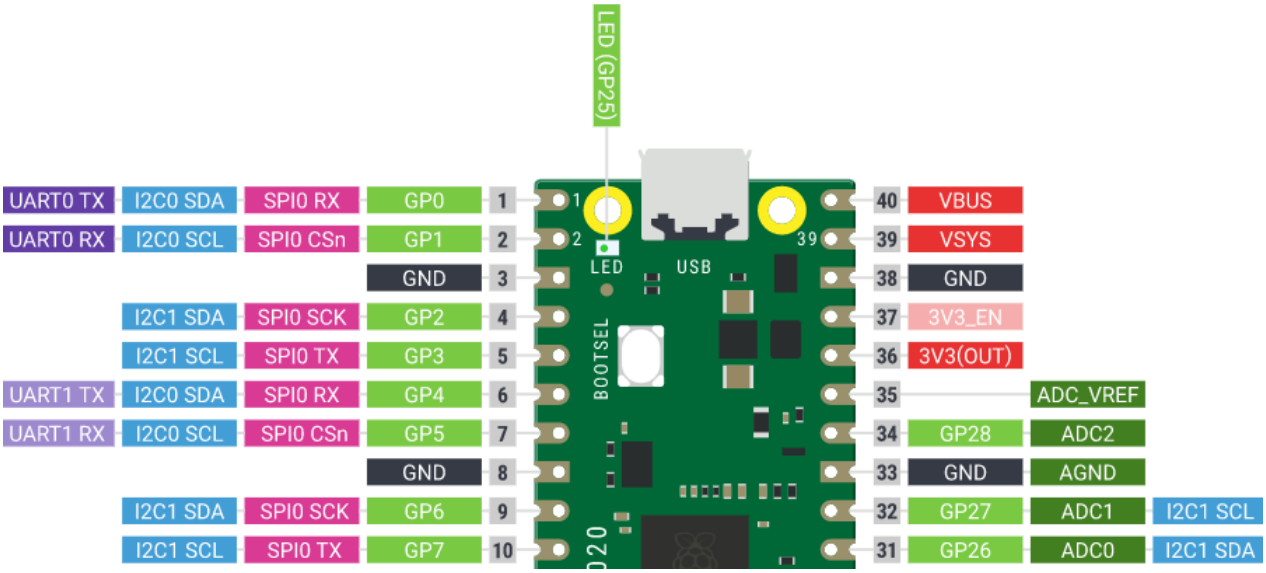
```
from machine import Pin
LED = Pin(25, Pin.OUT)
sw = Pin(16, Pin.IN, Pin.PULL_UP)   # 設定 GP16 為 sw 按鈕開關輸入腳，且自帶 pull high 電阻
while True:
    if(sw.value() == 0):    # 若 sw 輸入為 Low(按下接地)
        LED.value(1)        # 點亮 LED
    else:
        LED.value(0)        # 熄滅 LED1
```

類比輸入測試

※ A07_可變電阻控制內建 LED.py 程式碼如下：

```
from machine import Pin, ADC, PWM
import utime                # 導入時間相關類別
pwm = PWM(Pin(25))          # 設定 LED 為 PWM 輸出腳
pwm.freq(1000)              # 設定 PWM 頻率為 1000 Hz
adc = ADC(0)                # 設定連接到 ADC0(GP26)
factor = 3.3 / (65535)      # 電壓轉換因子
while True:
    reading = adc.read_u16()        # u16 代表讀取類比輸入值 16bit 無號整數
    pwm.duty_u16(reading)           # 將輸入值轉至 PWM 工作周期值
    vlot = reading * factor         # 將輸入值轉成電壓值
    print(vlot)                     # 將輸入電壓值列印至 Shell 區
    utime.sleep(0.1)                # 延時 0.1 秒
```

註：在 Pi Pico 上 GP1 到 GP28 除了當作一般數位輸出/輸入外，亦可作為 PWM 類比輸出點，但只有 GP26,27,28 才可以作為類比輸入點(ADC0,ADC1,ADC2)。另外晶片內部有一溫度感測元件作為 ADC3 類比輸入。Pi Pico 上有提供 3.3V 類比轉換參考電壓 ADC_VREF 及類比接地信號 AGND。在這個測試電路中，使用一個 10K 歐姆的半可變電阻(SVR)作為類比信號的模擬輸入，而隨意指定一個 LED 作為 PWM（模擬類比）輸出。上述範例是讀取 ADC0(GP26)的數值，再將其轉成 PWM 信號輸出至 LED，如此當調整 SVR 時 ADC0 讀到電壓大則 LED 越亮，反之則令 LED 變暗，相當於是類比調整 LED 亮度。




```
3.172703
3.081611
2.956731
2.731092
2.50948
2.41436
2.350712
2.241089
2.103319
2.005782
1.871234
1.790616
1.607677
1.37398
```

※ A08_讀取內建溫度感測器溫度.py 程式碼如下：

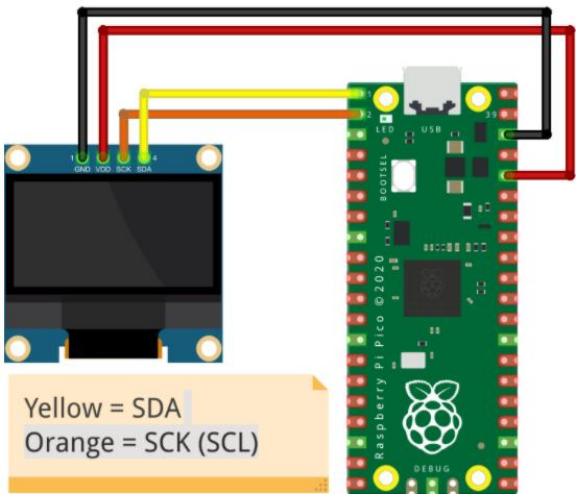
```
import machine
import utime
adc = machine.ADC(4)
conversion_factor = 3.3 / (65535)
while True:
    reading = adc.read_u16() * conversion_factor
    temperature = 25 - (reading - 0.706) / 0.001721
    out_string = "Temperature : " + str(temperature)
    print(out_string)
    utime.sleep(3)
```

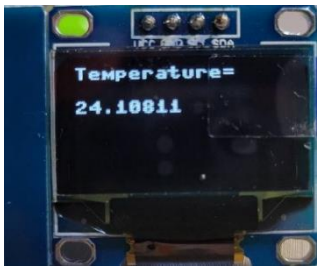
```
>>> %Run -c $EDITOR_CONTENT
 Temperature : 22.23554
 Temperature : 22.70368
 Temperature : 22.70368
 Temperature : 24.10811
 Temperature : 24.57626
```

註：溫度感測器是內建於系統晶片當中，當初目的應是怕晶片溫度過高而設計，所以是偵測 IC 表面溫度，而非周圍溫度，同時讀取 ADC(4)並轉換對應的數值，雖然不是非常準確，但溫度確實會有升降，應稱為『溫差感測』較適合，實際用手觸摸測試，溫度上升很明顯，但降溫則會慢慢下降顯示。

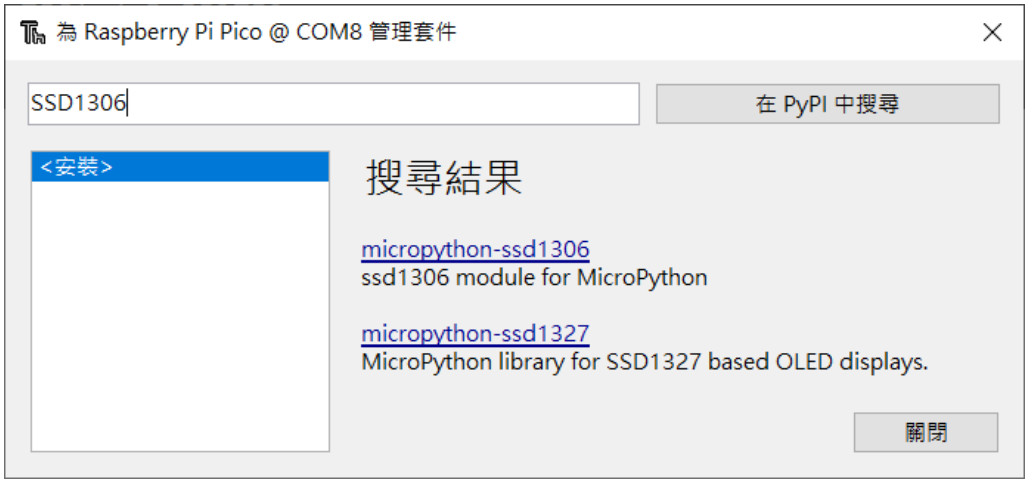
※ A09_用 OLED 顯示內建溫度感測器溫度.py 程式碼如下：

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import utime
adc = machine.ADC(4)
conversion_factor = 3.3 / (65535)
i2c = I2C(0, sda = Pin(0), scl = Pin(1), freq = 400000)
oled = SSD1306_I2C(128, 64, i2c)
while True:
    oled.fill(0)
    reading = adc.read_u16() * conversion_factor
    temperature = 25 - (reading - 0.706) / 0.001721
    out_string = "Temperature : " + str('%.3f' % temperature)
    print(out_string)
    oled.text("Temperature=", 0, 0)
    oled.text(str(temperature), 0, 20)
    oled.show()
    utime.sleep(3)
```



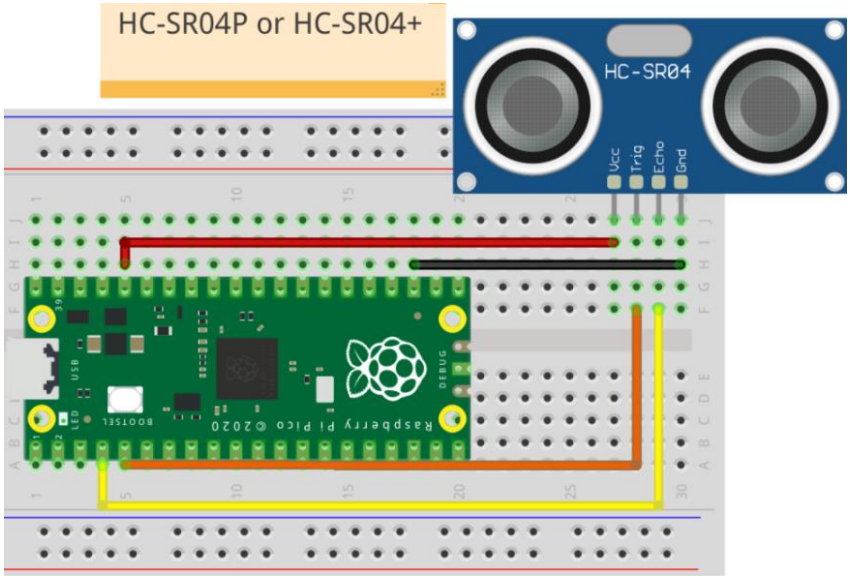


註：OLED 採 I2C 設計，若用另一組則為『i2c = I2C(1, sda = Pin(2), scl = Pin(3), freq = 400000)』，而 OLED 資料庫，依序按『工具 → 管理套件 → 輸入 SSD1306 → 在 PyPI 中搜尋 → 選擇 micropython-ssd1306 → 安裝』，方能執行上述程式碼。oled.fill(0)目的是清除螢幕。



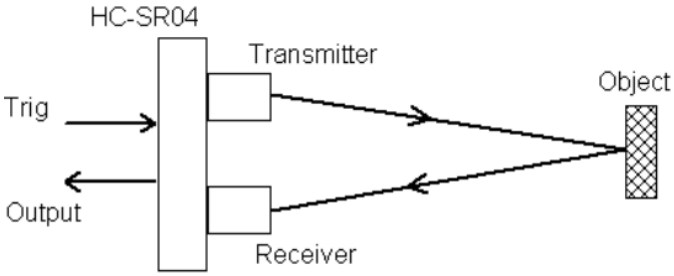
※ A10_超音波測距離.py 程式碼如下：

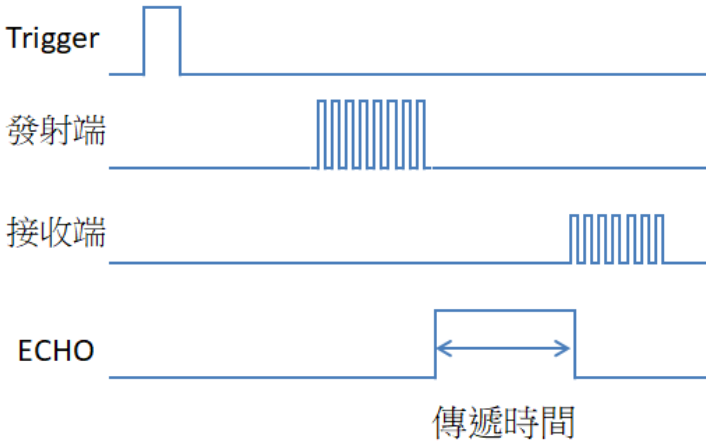
```
from machine import Pin
import utime
trigger = Pin(3, Pin.OUT)
echo = Pin(2, Pin.IN)
def ultra():
    trigger.low()
    utime.sleep_us(2)
    trigger.high()
    utime.sleep_us(5)
    trigger.low()
    while echo.value() == 0:
        signaloff = utime.ticks_us()
    while echo.value() == 1:
        signalon = utime.ticks_us()
    timepassed = signalon - signaloff
    distance = (timepassed * 0.0343) / 2
    print("The distance from object is ", distance, "cm")
while True:
    ultra()
    utime.sleep(1)
```




```
>>> %Run -c $EDITOR_CONTENT
 The distance from object is  2.17805 cm
 The distance from object is  1.90365 cm
 The distance from object is  2.9498 cm
 The distance from object is  81.58256 cm
 The distance from object is  43.61245 cm
 The distance from object is  13.6857 cm
```

註 1：超音波距離顯示於 OLED





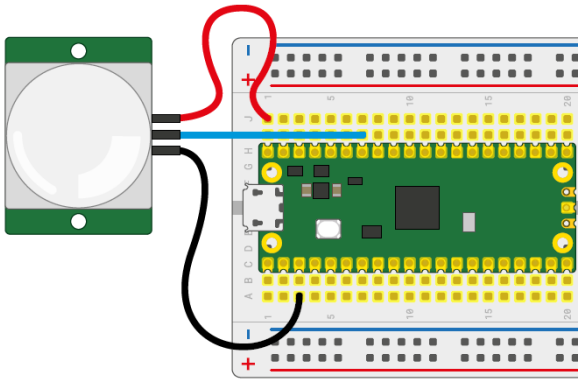
○ 動動腦將超音波距離顯示於 OLED

※ A11_PIR 人體感測器數位讀取.py 程式碼如下：

註：因為 PIR 工作電壓為 5V，需連接 Pin40 當電源，當有人經過為高電位，內建 LED 點亮。

PIR 全名是 Passive Infrared Sensor 被動式紅外線感測器的縮寫，工作電壓為 5V，在生活當中很多東西都會發射紅外線，例如燈泡、蠟燭，而人體也會發射紅外線，所以 PIR 便是利用人體發射出來的紅外線變化，來感應是否有人經過，一般用在防盜系統上，例如當有人入侵屋內便響警報；亦可當成自動照明裝置，例如走廊、樓梯間較少人走動場所，利用 PIR 有人就自動開燈照明，沒人經過就關燈，以節省電源。當有人進入其感應範圍則輸出高電位，模組指示燈會點亮，當人離開感應範圍會「自動延時」才輸出低電位，模組指示燈會熄滅，所以屬於「數位輸入」控制

```
from machine import Pin
LED = Pin(25, Pin.OUT)
PIR = Pin(28, Pin.IN, Pin.PULL_UP)
while True:
    if(PIR.value() == 1):
        LED.value(1)      # 點亮 LED
    else:
        LED.value(0)      # 熄滅 LED
```



A12_PIR 人體感測器中斷讀取.py 程式碼如下：

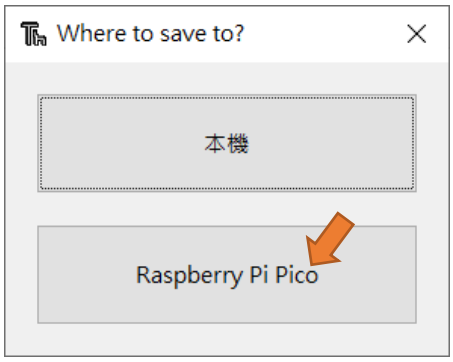
註：本程式是利用中斷請求(Interrupt Request)於背景執行，所以執行時，不可再同時開啟另一個程式的中斷，否則會出現下面錯誤，除非按 CTRL+C 或停止第一個程式，toggle 是切換的意思，LED 會閃爍，RISING 是上升的意思，平時為 0，有人經過為 1，所以當 0 到 1 時代表有人經過，優點是不會一直傳送有人移動的通知。

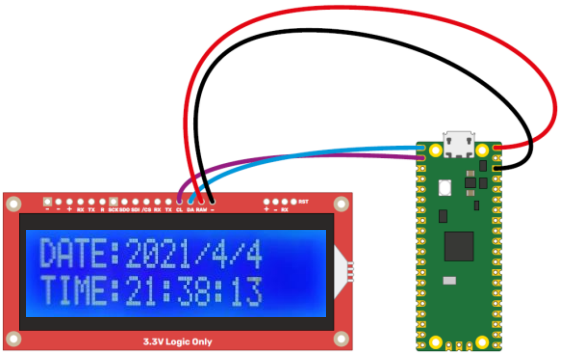
```
互動環境 (Shell)

Device is busy or does not respond. Your options:

  - wait until it completes current work;
  - use Ctrl+C to interrupt current work;
  - use Stop/Restart to interrupt more and enter REPL.
```

```
import machine
import utime
PIR = machine.Pin(28, machine.Pin.IN, machine.Pin.PULL_DOWN)
led = machine.Pin(25, machine.Pin.OUT)
def pir_handler(pin):
    utime.sleep_ms(100)
    if pin.value():
        print("ALARM! Motion detected!")
        for i in range(50):
            led.toggle()
            utime.sleep_ms(100)
PIR.irq(trigger = machine.Pin.IRQ_RISING, handler = pir_handler)
```

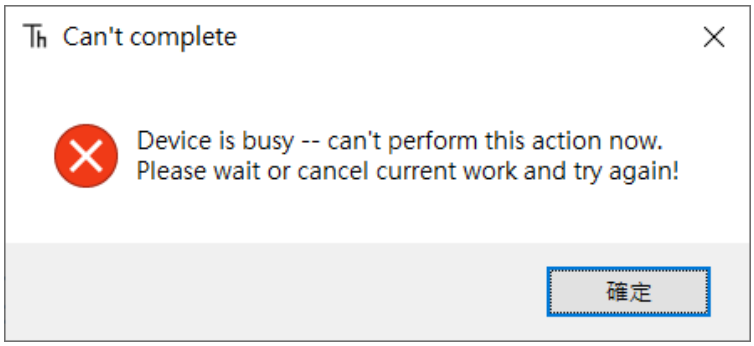
A13_I2C LCD 顯示系統時間.py 程式碼如下：

註 1：執行前須先將 lcd_api.py 與 pico_i2c_lcd.py 放在 Pico 的裡面，另存新檔如左下圖(.py 要輸入)，才能正確執行本程式，否則會出現找不到 lcd_api.py 錯誤訊息。硬體連接 pin1 的 SDA 與 pin2 的 SDA，或其他組 I2C 均可，但腳位設定須改為『i2c = I2C(1, sda = machine.Pin(2), scl = machine.Pin(3), freq = 400000)』





註 2：若要讓開發板一開機就執行該程式，請將左上圖檔案命名成 **main.py 或 boot.py，但時間會跑掉。**



需先按 CTRL+C 停止程式才可寫入新程式。

```
import utime
import machine
from machine import I2C
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd
I2C_ADDR          = 0x27
I2C_NUM_ROWS = 2
I2C_NUM_COLS = 16
def test_main():
    print("Running test_main")
    i2c = I2C(0, sda = machine.Pin(0), scl = machine.Pin(1), freq = 400000)
    lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)
    lcd.putstr("Pi Pico Works!")
    utime.sleep(2)
    lcd.clear()
    count = 0
    while True:
        lcd.clear()
        time = utime.localtime()
        lcd.move_to(0, 0)
        lcd.putstr("DATE:{year}/{month}/{day}".format(year=str(time[0]),month=str(time[1]),
day=str(time[2])))
        lcd.move_to(0, 1)
        lcd.putstr("TIME:{HH}:{MM}:{SS}".format(HH=str(time[3]), MM=str(time[4]), SS=str(time[5])))
        if count % 10 == 0:
            print("Turning cursor on")
            lcd.show_cursor()
        if count % 10 == 1:
            print("Turning cursor off")
            lcd.hide_cursor()
```

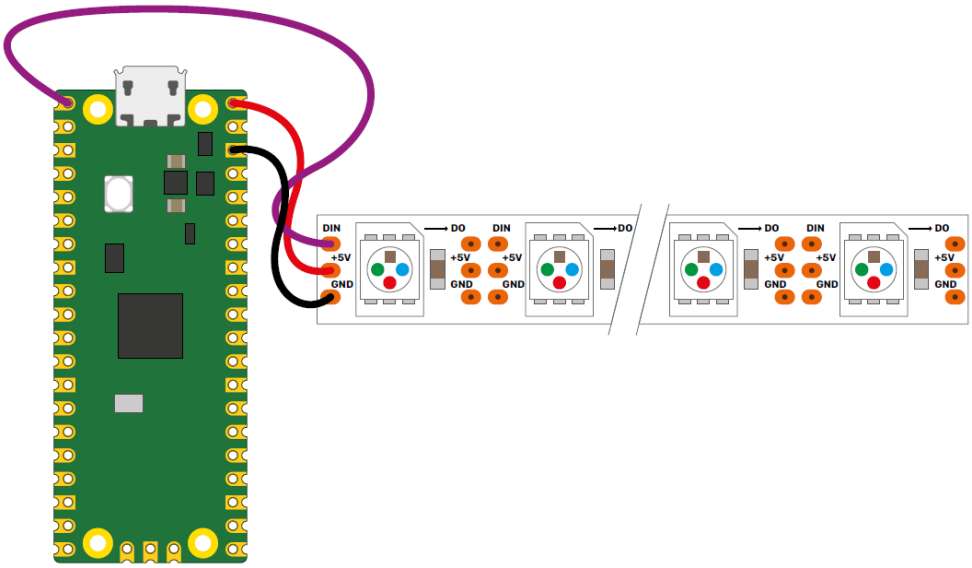
```
30            if count % 10 == 2:
31                print("Turning blink cursor on")
32                lcd.blink_cursor_on()
33            if count % 10 == 3:
34                print("Turning blink cursor off")
35                lcd.blink_cursor_off()
36            if count % 10 == 4:
37                print("Turning backlight off")
38                lcd.backlight_off()
39            if count % 10 == 5:
40                print("Turning backlight on")
41                lcd.backlight_on()
42            if count % 10 == 6:
43                print("Turning display off")
44                lcd.display_off()
45            if count % 10 == 7:
46                print("Turning display on")
47                lcd.display_on()
48            if count % 10 == 8:
49                print("Filling display")
50                lcd.clear()
51                string = ""
52                for x in range(32, 32+I2C_NUM_ROWS*I2C_NUM_COLS):
53                    string += chr(x)
54                lcd.putstr(string)
55            count += 1
56            utime.sleep(2)
57
58    #if __name__ == "__main__":
59    test_main()
```

```
互動環境 (Shell)
MicroPython v1.14 on 2021-04-03; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
 Running test_main
 Turning cursor on
 Turning cursor off
 Turning blink cursor on
 Turning blink cursor off
 Turning backlight off
 Turning backlight on
 Turning display off
 Turning display on
 Filling display
 Traceback (most recent call last):
   File "<stdin>", line 77, in <module>
   File "<stdin>", line 37, in test_main
   File "lcd_api.py", line 66, in clear
   File "pico_i2c_lcd.py", line 58, in hal_write_command
 OSError: [Errno 5] EIO
```

註：最下面的錯誤是因為在執行過程中，突然移除，沒有偵測到 I2C LCD。

A14_串列式全彩 LED.py 程式碼如下：

註：WS2812 串列式全彩 LEDs 與溫溼度 DHT11 都是採用 Programmable Input and Output(PIO)技巧。




```
import array, utime
from machine import Pin
import rp2
from rp2 import PIO, StateMachine, asm_pio
NUM_LEDS = 12
@asm_pio(sideset_init = PIO.OUT_LOW, out_shiftdir = PIO.SHIFT_LEFT, autopull = True, pull_thresh = 24)
def ws2812():
    T1 = 2
    T2 = 5
    T3 = 3
    label("bitloop")
    out(x, 1)                .side(0) [T3 - 1]
    jmp(not_x, "do_zero") .side(1) [T1 - 1]
    jmp("bitloop")           .side(1) [T2 - 1]
    label("do_zero")
    nop()                    .side(0) [T2 - 1]
# Create the StateMachine with the ws2812 program, outputting on Pin(12).
sm = StateMachine(0, ws2812, freq = 8000000, sideset_base = Pin(12))
# Start the StateMachine, it will wait for data on its FIFO.
sm.active(1)
# Display a pattern on the LEDs via an array of LED RGB values.
ar = array.array("I", [0 for _ in range(NUM_LEDS)])
print("blue")
for j in range(0, 255):
    for i in range(NUM_LEDS):
        ar[i] = j
    sm.put(ar,8)
    utime.sleep_ms(10)
print("red")
```

```
for j in range(0, 255):
    for i in range(NUM_LEDS):
        ar[i] = j << 8
    sm.put(ar,8)
    utime.sleep_ms(10)
print("green")
for j in range(0, 255):
    for i in range(NUM_LEDS):
        ar[i] = j << 16
    sm.put(ar,8)
    utime.sleep_ms(10)
print("white")
for j in range(0, 255):
    for i in range(NUM_LEDS):
        ar[i] = (j << 16) + (j << 8) + j
    sm.put(ar,8)
    utime.sleep_ms(10)
ar = array.array("I", [0 for _ in range(NUM_LEDS)])
sm.put(ar,8)
```

A15_串列式全彩 LED-2.py 程式碼如下：

```
import array, time
from machine import Pin
import rp2
# Configure the number of WS2812 LEDs.
NUM_LEDS = 12
PIN_NUM = 12
brightness = 0.5
@rp2.asm_pio(sideset_init = rp2.PIO.OUT_LOW, out_shiftdir = rp2.PIO.SHIFT_LEFT,
autopull = True, pull_thresh=24)

def ws2812():
    T1 = 2
    T2 = 5
    T3 = 3
    wrap_target()
    label("bitloop")
    out(x, 1)                .side(0)    [T3 - 1]
    jmp(not_x, "do_zero")    .side(1)    [T1 - 1]
    jmp("bitloop")           .side(1)    [T2 - 1]
    label("do_zero")
    nop()                    .side(0)    [T2 - 1]
    wrap()
```

```
# Create the StateMachine with the ws2812 program, outputting on pin
sm = rp2.StateMachine(0, ws2812, freq=8_000_000, sideset_base=Pin(PIN_NUM))
# Start the StateMachine, it will wait for data on its FIFO.
sm.active(1)
# Display a pattern on the LEDs via an array of LED RGB values.
ar = array.array("I", [0 for _ in range(NUM_LEDS)])

def pixels_show():
    dimmer_ar = array.array("I", [0 for _ in range(NUM_LEDS)])
    for i,c in enumerate(ar):
        r = int(((c >> 8) & 0xFF) * brightness)
        g = int(((c >> 16) & 0xFF) * brightness)
        b = int((c & 0xFF) * brightness)
        dimmer_ar[i] = (g << 16) + (r << 8) + b
    sm.put(dimmer_ar, 8)
    time.sleep_ms(10)

def pixels_set(i, color):
    ar[i] = (color[1] << 16) + (color[0] << 8) + color[2]

def pixels_fill(color):
    for i in range(len(ar)):
        pixels_set(i, color)

def color_chase(color, wait):
    for i in range(NUM_LEDS):
        pixels_set(i, color)
        time.sleep(wait)
        pixels_show()
    time.sleep(0.2)

def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)
```

```
def rainbow_cycle(wait):
    for j in range(255):
        for i in range(NUM_LEDS):
            rc_index = (i * 256 // NUM_LEDS) + j
            pixels_set(i, wheel(rc_index & 255))
        pixels_show()
        time.sleep(wait)

BLACK = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)
WHITE = (255, 255, 255)
COLORS = (BLACK, RED, YELLOW, GREEN, CYAN, BLUE, PURPLE, WHITE)

print("fills")
for color in COLORS:
    pixels_fill(color)
    pixels_show()
    time.sleep(0.2)

print("chases")
for color in COLORS:
    color_chase(color, 0.01)
print("rainbow")
rainbow_cycle(0)
```

A16_讀取溫濕度 DHT11.py 程式碼如下： (註：利用 GPIO11 讀取溫濕度資料)

```
import utime
import rp2
from rp2 import PIO, asm_pio
from machine import Pin

@asm_pio(set_init = (PIO.OUT_HIGH),autopush = True, push_thresh = 8)    #output one byte at a time
def DHT11():
    #drive output low for at least 20ms
    set(pindirs,1)                  #set pin to output
    set(pins,0)                      #set pin low
    set(y,31)                        #prepare countdown, y*x*100cycles
    label ('waity')
    set(x,31)
    label ('waitx')
    nop() [25]
    nop() [25]
    nop() [25]
    nop() [25]                       #wait 100cycles
    jmp(x_dec,'waitx')              #decrement x reg every 100 cycles
    jmp(y_dec,'waity')              #decrement y reg every time x reaches zero

    #begin reading from device
    set(pindirs,0)                   #set pin to input
    wait(1,pin,0)                    #check pin is high before starting
    wait(0,pin,0)
    wait(1,pin,0)
    wait(0,pin,0)                    #wait for start of data

    #read databit
    label('readdata')
    set(x,20)                        #reset x register to count down from 20
    wait(1,pin,0)                    #wait for high signal
    label('countdown')
    jmp(pin,'continue')              #if pin still high continue counting
    #pin is low before countdown is complete - bit '0' detected
    set(y,0)
    in_(y, 1)                        #shift '0' into the isr
    jmp('readdata')                  #read the next bit

    label('continue')
    jmp(x_dec,'countdown')           #decrement x reg and continue counting if x!=0
    #pin is still high after countdown complete - bit '1' detected
```

```
43          set(y,1)
44          in_(y, 1)                       #shift one bit into the isr
45          wait(0,pin,0)                   #wait for low signal (next bit)
46          jmp('readdata')                 #read the next bit
47
48  #main program
49  dht_data = Pin(11, Pin.IN, Pin.PULL_UP) #connect GPIO 11 to 'out' on DHT11
50  sm = rp2.StateMachine(1)                #create empty state machine
51  utime.sleep(2)                          #wait for DHT11 to start up
52
53  while True:
54          print('reading')
55          data=[]
56          total = 0
57          sm.init(DHT11, freq = 1600000, set_base = dht_data, in_base = dht_data, jmp_pin = dht_data)
58          #start state machine
59          #state machine frequency adjusted so that PIO countdown during 'readdata' ends somewhere between the
60          #duration of a '0' and a '1' high signal
61          sm.active(1)
62          for i in range(5):              #data should be 40 bits (5 bytes) long
63                  data.append(sm.get())   #read byte
64          print("data: " + str(data))
65          #check checksum (lowest 8 bits of the sum of the first 4 bytes)
66          for i in range(4):
67                  total = total + data[i]
68          if((total & 255) == data[4]):
69                  humidity = data[0]              #DHT11 provides integer humidity (no decimal part)
70                  temperature = (1 – 2 * (data[2]    >>    7) )*(data[2] & 0x7f)
71                  #DHT11 provides signed integer temperature (no decimal part)
72                  print("Humidity: %d%%, Temp: %dC" % (humidity, temperature))
73          else:
74                  print("Checksum: failed")
75          utime.sleep_ms(500)
```

互動環境 (Shell)

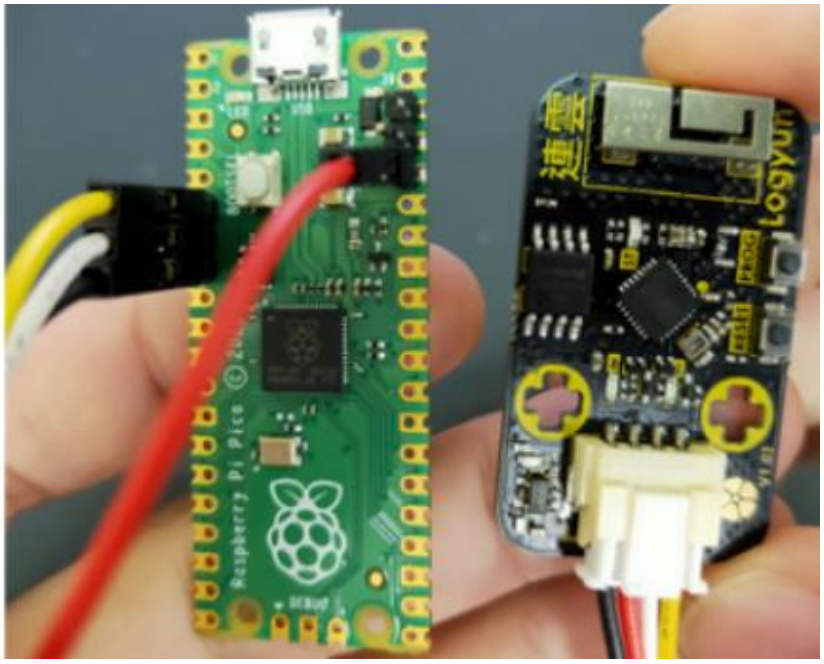
```
MicroPython v1.14 on 2021-04-03; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
 reading
 data: [52, 0, 25, 0, 77]
 Humidity: 52%, Temp: 25C
 reading
 data: [50, 0, 25, 0, 75]
 Humidity: 50%, Temp: 25C
```

A17_LCD 顯示 DHT11 溫濕度值.py 程式碼如下：

```
import utime
import rp2
import machine
from rp2 import PIO, asm_pio
from machine import Pin
from machine import I2C
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd
I2C_ADDR          = 0x27
I2C_NUM_ROWS = 2
I2C_NUM_COLS = 16
i2c = I2C(0, sda = machine.Pin(0), scl = machine.Pin(1), freq = 400000)
lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)
lcd.putstr("DHT11 is Works!")
utime.sleep(2)
lcd.clear()
@asm_pio(set_init = (PIO.OUT_HIGH), autopush = True, push_thresh = 8)
def DHT11():
    set(pindirs, 1)
    set(pins, 0)
    set(y,31)
    label ('waity')
    set(x,31)
    label ('waitx')
    nop() [25]
    nop() [25]
    nop() [25]
    nop() [25]
    jmp(x_dec,'waitx')
    jmp(y_dec,'waity')
    set(pindirs,0)
    wait(1,pin,0)
    wait(0,pin,0)
    wait(1,pin,0)
    wait(0,pin,0)
    label('readdata')
    set(x,20)
    wait(1,pin,0)
    label('countdown')
    jmp(pin,'continue')
    set(y,0)
    in_(y, 1)
```

```
43        jmp('readdata')
44        label('continue')
45        jmp(x_dec,'countdown')
46        set(y,1)
47        in_(y, 1)
48        wait(0,pin,0)
49        jmp('readdata')
50  dht_data = Pin(11, Pin.IN, Pin.PULL_UP)
51  sm = rp2.StateMachine(1)
52  utime.sleep(2)
53  while True:
54        data=[]
55        total = 0
56        sm.init(DHT11, freq = 1600000, set_base = dht_data, in_base = dht_data, jmp_pin = dht_data)
57        sm.active(1)
58        for i in range(5):
59              data.append(sm.get())
60        print("data: " + str(data))
61        for i in range(4):
62              total = total + data[i]
63        if((total & 255) == data[4]):
64              humidity = data[0]
65              temperature=(1 - 2 * (data[2] >> 7) )*(data[2] & 0x7f)
66              lcd.move_to(0, 0)
67              lcd.putstr("Temp = %d C" % (temperature))
68              lcd.move_to(0, 1)
69              lcd.putstr("Humi = %d %%" % (humidity))
70        else:
71              lcd.move_to(0, 0)
72              lcd.putstr("DHT11 failed...")
73        utime.sleep_ms(500)
```

※ 搭配『Logyun 連雲』方可連到網路 (參考網址 http://www.circuspi.com/index.php/2021/02/24/logyun-tutorial/#Raspberry-Pi-Pico-%E4%BD%BF%E7%94%A8%E6%96%B9%E6%B3%95)
連接線的 Grove 端照樣插在 Logyun 連雲，另一端杜邦母線連接在 Pi Pico 上，連接 RX 的黃色線接在 Pi Pico 的 GPIO4，連接 TX 的白色線接在 Pi Pico 的 GPIO5，GND 的黑色線接在 Pi Pico 任一 GND，連接 VCC 的紅色線接在 Pi Pico 的 3V3(OUT)。如下圖所示：



※ B01 連雲查詢 IP.py 程式碼如下：

```
from machine import UART
from time import sleep
uart = UART(1, 115200)   # 使用 UART1 的 GP4 與 GP5
uart.write("WifiConnect(TSSH, 12345678)")
print(uart.readline())
while True:
    uart.write("WifiCheck()")
    check = uart.readline()
    if check == b'ok\n':        # 因為數據返回的資料類型是 byte = b
        uart.write("WifiLocalIP()")
        ip = str(uart.readline())[2:-3]
        print(ip)
    sleep(1)
```

執行結果如下：

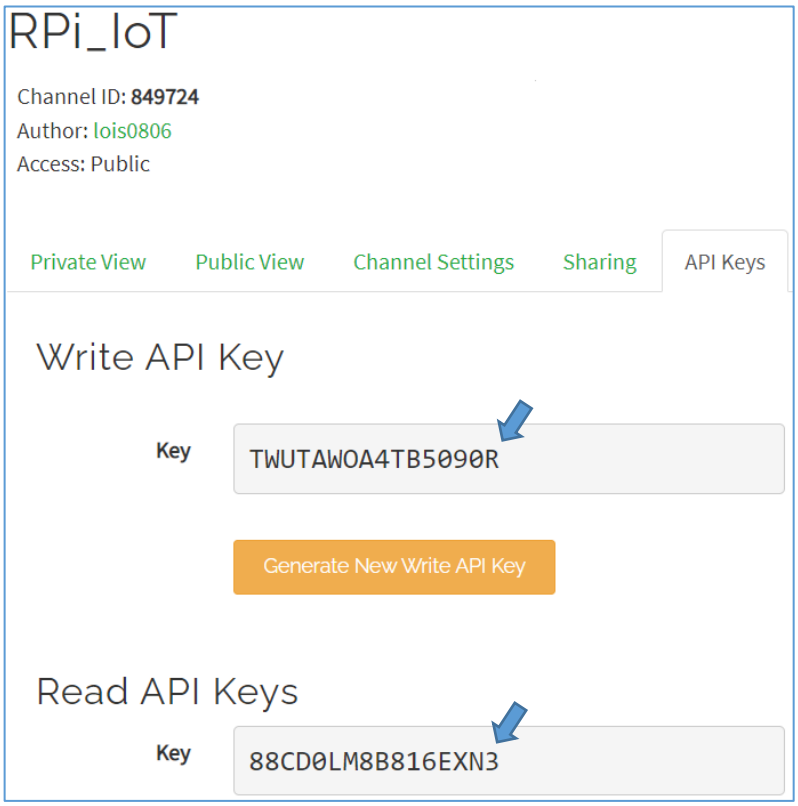
```
互動環境 (Shell)
MicroPython v1.14 on 2021-02-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
 b'ok\n'
 192.168.28.164
 192.168.28.164
```

連網功能
WifiConnect(Wi-Fi名稱,密碼)
WifiCheck()
WifiLocalIP()

註：WifiLocalIP() 的返回值實際為 b’192.168.28.164\n’，利用 str(uart.readline())[2:-3] 就可以將 b’192.168.28.164\n’ 轉變成 192.168.28.164，其中[2:-3]代表抓取第 2 個字元到倒數第 3 個字元中間字串，負號是表示從後面倒數過來。

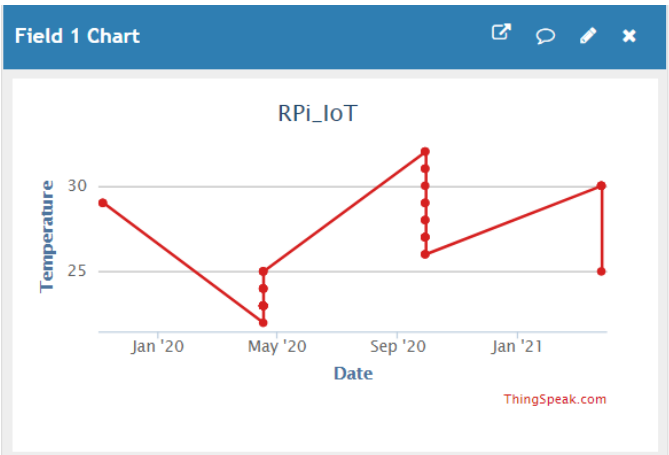
※ B02 連雲 ThingSpeak 寫入.py 程式碼如下：

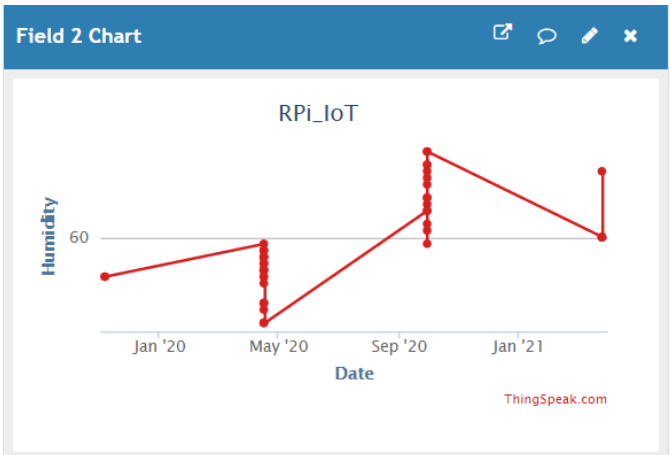
```
from machine import UART
from time import sleep
uart = UART(1, 115200)
uart.write("WifiConnect(TSSH, 12345678)")
print(uart.readline())
while True:
    uart.write("ThingSpeakWrite(TWUTAWOA4TB5090R, 25, 70)")
    print(uart.readline())
    sleep(10)
```



資料存取功能
ThingSpeakWrite(Write API Keys,資料1,資料2,...,資料8)
ThingSpeakRead(Channel ID,Read API,資料表格)
GoogleSheetWrite(試算表ID,資料1,資料2,...,資料10)
GoogleSheetRead(試算表ID,資料位址)

執行結果如下：





※ B03 連雲 ThingSpeak 讀取.py 程式碼如下：

(註：讀取 ThingSpeak 資料上有一個限制，只允許抓取最新的一筆資料)

```
from machine import UART
from time import sleep
uart = UART(1, 115200)
uart.write("WifiConnect(TSSH, 12345678)")
print(uart.readline())
while True:
    uart.write("ThingSpeakRead(849724, 88CD0LM8B816EXN3, 1)")   # 讀取欄位 1 最新資料
    print(str(uart.readline())[2:-3])
    uart.write("ThingSpeakRead(849724, 88CD0LM8B816EXN3, 2)")   # 讀取欄位 2 最新資料
    print(str(uart.readline())[2:-3])
    sleep(10)
```

執行結果如下：

```
互動環境 (Shell)

MicroPython v1.14 on 2021-02-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
 b'ok\n'
 25
 70
 25
 70
```

Read API Keys

Key 88CD0LM8B816EXN3

※ B04 連雲 Google 工作表資料寫入.py 程式碼如下：

註：GoogleSheetWrite(試算表 ID,資料 1,資料 2,…,資料 10)」，這能將資料上傳到指定的 Google Sheet ，最多傳送 10 筆資料，輸入的參數皆以逗號隔開，字串中間不允許多餘的空格，無法指定哪個工作表。

```
from machine import UART
from time import sleep
uart = UART(1, 115200)
uart.write("WifiConnect(TSSH, 12345678)")
print(uart.readline())
while True:
    uart.write("GoogleSheetWrite(1CEQckux79olroeW8Pws6zze2HI7Auy6eO0yizRiEWO0,12,15,20,88)")
    print(uart.readline())
    sleep(5)
```

執行結果如下：

	A	B	C	D	E
1	2021/03/27 6:47:43	12	15	20	88
2	2021/03/27 6:47:50	12	15	20	88
3	2021/03/27 6:47:57	12	15	20	88
4	2021/03/27 6:48:04	12	15	20	88
5	2021/03/27 6:48:12	12	15	20	88

※ B05 連雲 Google 工作表資料讀取.py 程式碼如下：

註：GoogleSheetRead( )字串中間不允許多餘的空格。

```
from machine import UART
from time import sleep
uart = UART(1, 115200)
uart.write("WifiConnect(TSSH, 12345678)")
print(uart.readline())
while True:
    uart.write("GoogleSheetRead(1CEQckux79olroeW8Pws6zze2HI7Auy6eO0yizRiEWO0,C3)")
    print(str(uart.readline())[2:-3])
    sleep(5)
```

執行結果如下：

```
MicroPython v1.14 on 2021-02-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
 b'ok\n'
 15
```

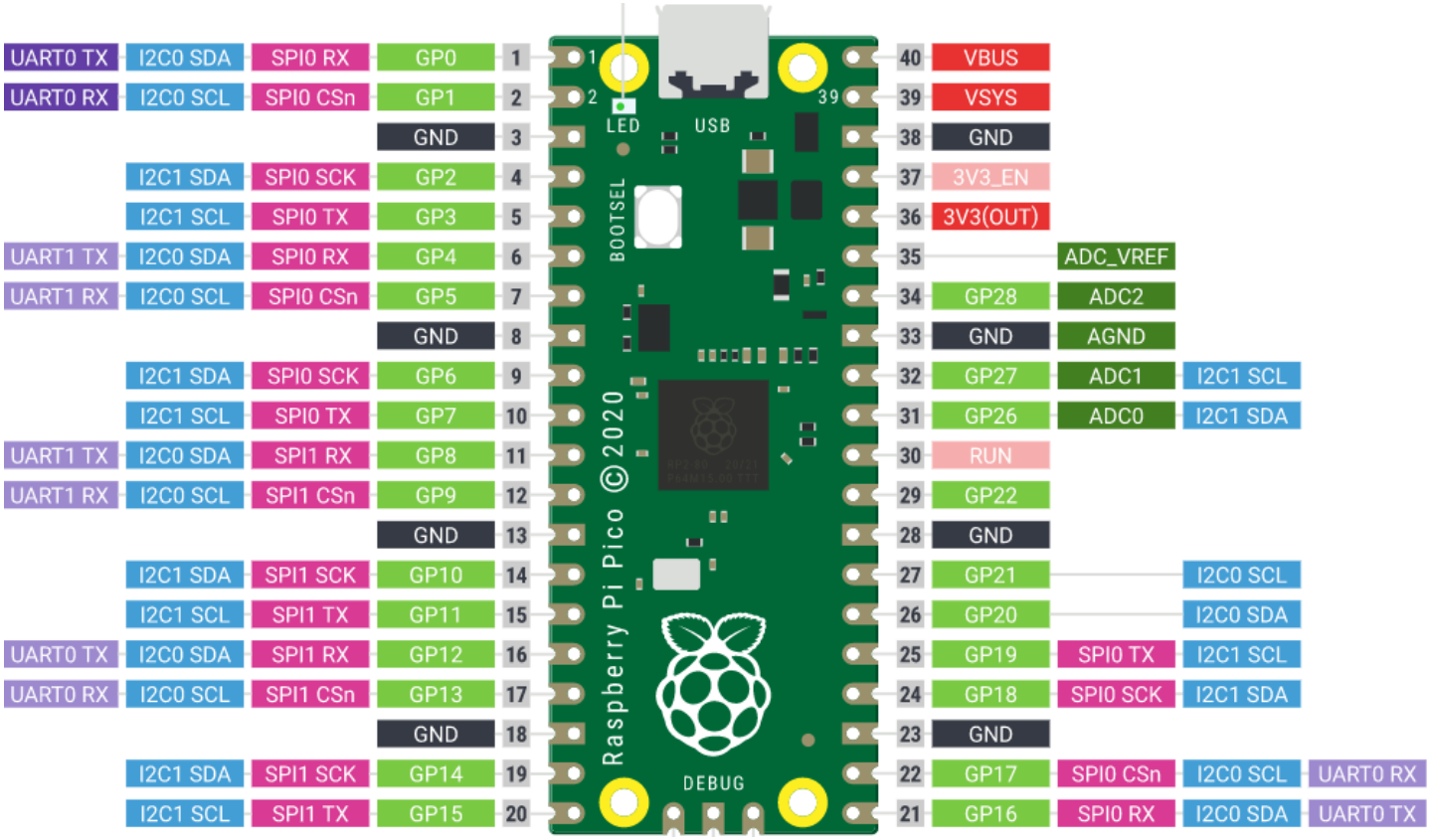
※ B05 連雲 Google 工作表資料讀取.py 程式碼如下：

指令名稱	功能說明	輸入參數	回傳訊息
MQTTConnect(broker網址,通訊埠,自訂ID,使用者名稱,使用者密碼,訂閱名稱)	透過MQTT連線指定broker與訂閱主題	broker網址 通訊埠 自訂ID 使用者名稱(非必要但需保留) 使用者密碼(非必要但需保留) 訂閱名稱	回傳:ok
MQTTPublish(訂閱名稱,發佈訊息)	發佈訊息至訂閱的主題	訂閱名稱 發佈訊息	回傳:ok

註：MQTTConnect()字串中間不允許多餘的空格，連線後，我們將經由「MQTTPublish(訂閱名稱,發佈訊息)」這條指令，發佈我們想發送的訊息，將以 3 秒的間隔不停發送 0 與 1 的訊息。

```
from machine import UART
from time import sleep
uart = UART(1, 115200)
uart.write("WifiConnect(TSSH, 12345678)")
print(uart.readline())
uart.write("MQTTConnect(broker.hivemq.com,1883,0955974622,,,0955974622/pico)")
print(uart.readline())
while True:
    uart.write("MQTTPublish(0955974622/pico, 1)")
    print(uart.readline())
    sleep(3)
    uart.write("MQTTPublish(0955974622/pico, 0)")
    print(uart.readline())
    sleep(3)
```

```
MicroPython v1.14 on 2021-02-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
 b'ok\n'
 b'ok\n'
 b'ok\n'
 b'ok\n'
 b'ok\n'
 b'ok\n'
```



▼ 建議腳位使用如下，增加 3.3V/5V 電源開關切換，採 SVGG 母座設計。

pin	GPIO 腳位	功能	pin	GPIO 腳位	功能
1	GP0 (I2C0 SDA)	OLED	40	VBUS	輸出 5V 電源(接 SW)
2	GP1 (I2C0 SCL)	OLED	39	VSYS	外接 2~5V 電源(接 SW)
3	GND		38	GND	
4	GP2	超音波 echo	37	3V3(EN)	
5	GP3	超音波 trigger	36	3V3(OUT)	輸出 3.3V 電源
6	GP4 (UART1 Tx)	連雲 Rx	35	ADC_VREF	外部精密電壓參考
7	GP5 (UART1 Rx)	連雲 Tx	34	GP28 (ADC2)	類比輸入感測(如火焰)
8	GND		33	GND	AGND
9	GP6(I2C1 SDA)	LCD(5V)	32	GP27 (ADC1)	搖桿 VRy(類比輸入)
10	GP7(I2C1 SCL)	LCD(5V)	31	GP26 (ADC0)	搖桿 VRx(類比輸入)
11	GP8	PIR 人體移動偵測(5V)	30	RUN	啟用或禁用 RP2040
12	GP9	SERVO1(連續型)	29	GP22	搖桿按鈕(設提升 R)
13	GND		28	GND	
14	GP10	SERVO2(不連續型)	27	GP21	繼電器
15	GP11	DHT11(5V)	26	GP20	無段按鈕開關
16	GP12	串列全彩 LED(5V)	25	GP19	有源 Buzzer
17	GP13	RED LED	24	GP18	RGB LED
18	GND		23	GND	
19	GP14	YELLOW LED	22	GP17	RGB LED
20	GP15	GREEN LED	21	GP16	RGB LED

By:新北市泰山高中 電子科 陳致中 110/4/7