

TrafficLight1

```
from machine import Pin
from utime import sleep
led_red = Pin(18, Pin.OUT)
led_amber = Pin(17, Pin.OUT)
led_green = Pin(16, Pin.OUT)
while True:
    led_red.value(1)
    sleep(2)
    led_red.value(0)
    led_amber.value(1)
    sleep(1)
    led_amber.value(0)
    led_green.value(1)
    sleep(2)
    led_green.value(0)
    led_amber.value(1)
    sleep(1)
    led_amber.value(0)
```


新增第一顆按鈕

```
button_1 = Pin(15, Pin.IN, Pin.PULL_UP)
```

#button_1使用上拉電阻,輸入pin號為15,另一端接GND

為何要上拉電阻?

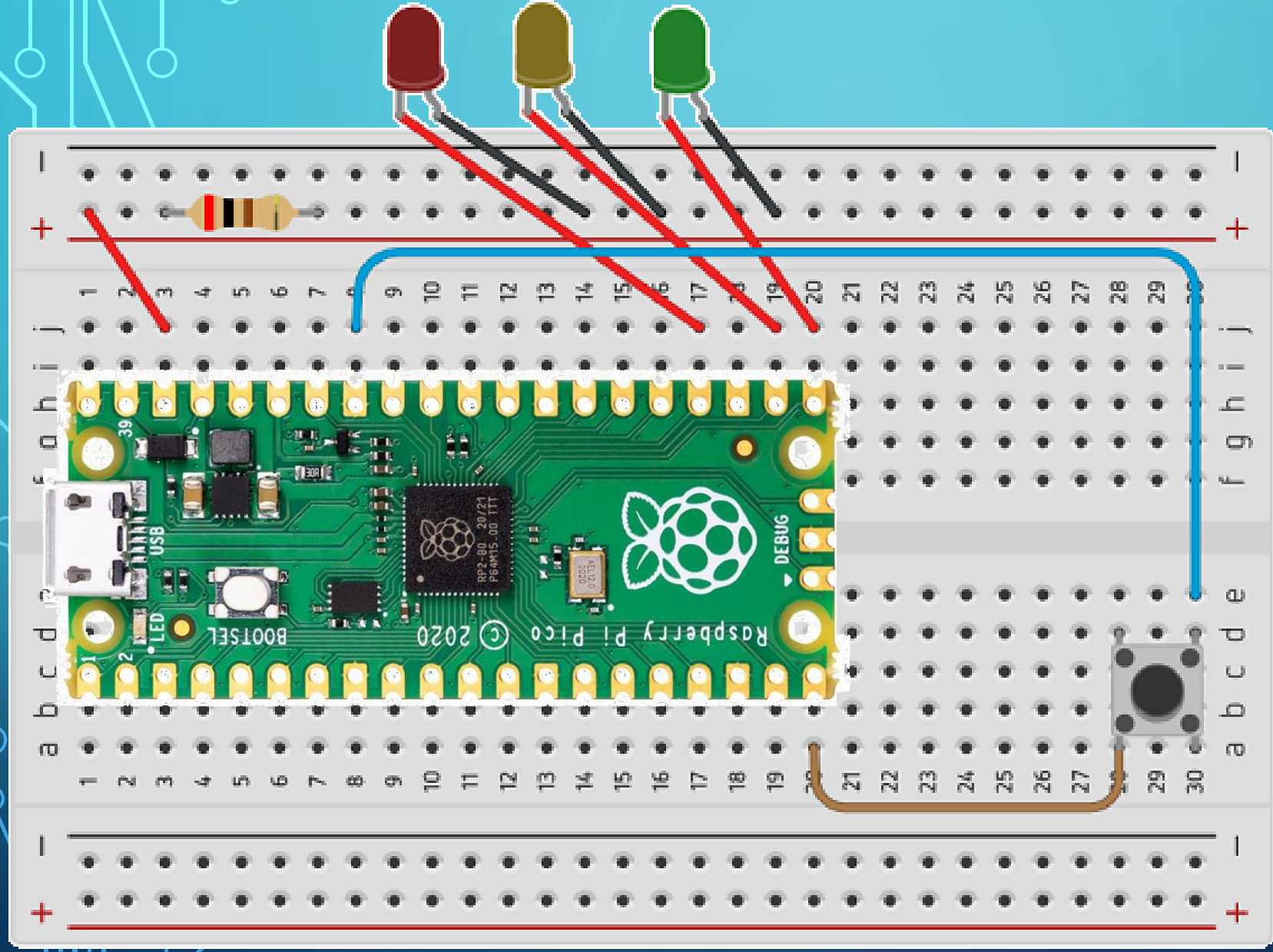
這是因為當 MCU 的 GPIO 某 pin 被設定為 INPUT mode 時,是在輸入高阻抗(input impedance)狀態,意思是相當於另一端有串接個超級大的電阻擋住,那意味著 pin 在空接時就等於沒有連接到任何電路,此時用digitalRead(pin)去讀取它則常常因受到環境雜訊的影響,有時讀取到 HIGH 有時卻讀取到 LOW,就是所謂的 floating 狀態!

**這樣按鈕沒按下也可能被誤判為有按下
為了確保它在穩定的狀態,必須接個上拉電阻或下拉電阻!**

為何要上拉電阻?

這是因為當 MCU 的 GPIO 某 pin 被設定為 INPUT mode 時,是在輸入高阻抗(input impedance)狀態,意思是相當於另一端有串接個超級大的電阻擋住,那意味著 pin 在空接時就等於沒有連接到任何電路,此時用digitalRead(pin)去讀取它則常常因受到環境雜訊的影響,有時讀取到 HIGH 有時卻讀取到 LOW, 就是所謂的 floating 狀態!

**這樣按鈕沒按下也可能被誤判為有按下
為了確保它在穩定的狀態, 必須接個上拉電阻或下拉電阻!**



按鈕範例

#按鈕使用範例 : button_01.py

```
from machine import Pin
```

```
from utime import sleep
```

```
led_red = Pin(18, Pin.OUT)
```

```
led_amber = Pin(17, Pin.OUT)
```

```
led_green = Pin(16, Pin.OUT)
```

```
button_1 = Pin(15, Pin.IN, Pin.PULL_UP)
```

#本例使用上拉電阻,輸入pin號為15,另一端接GND

```
while True:
```

```
    if button_1.value() == 0:
```

```
        for i in range(3):
```

```
            led_amber.value(1)
```

```
            sleep(0.5)
```

```
            led_amber.value(0)
```

```
            sleep(0.5)
```

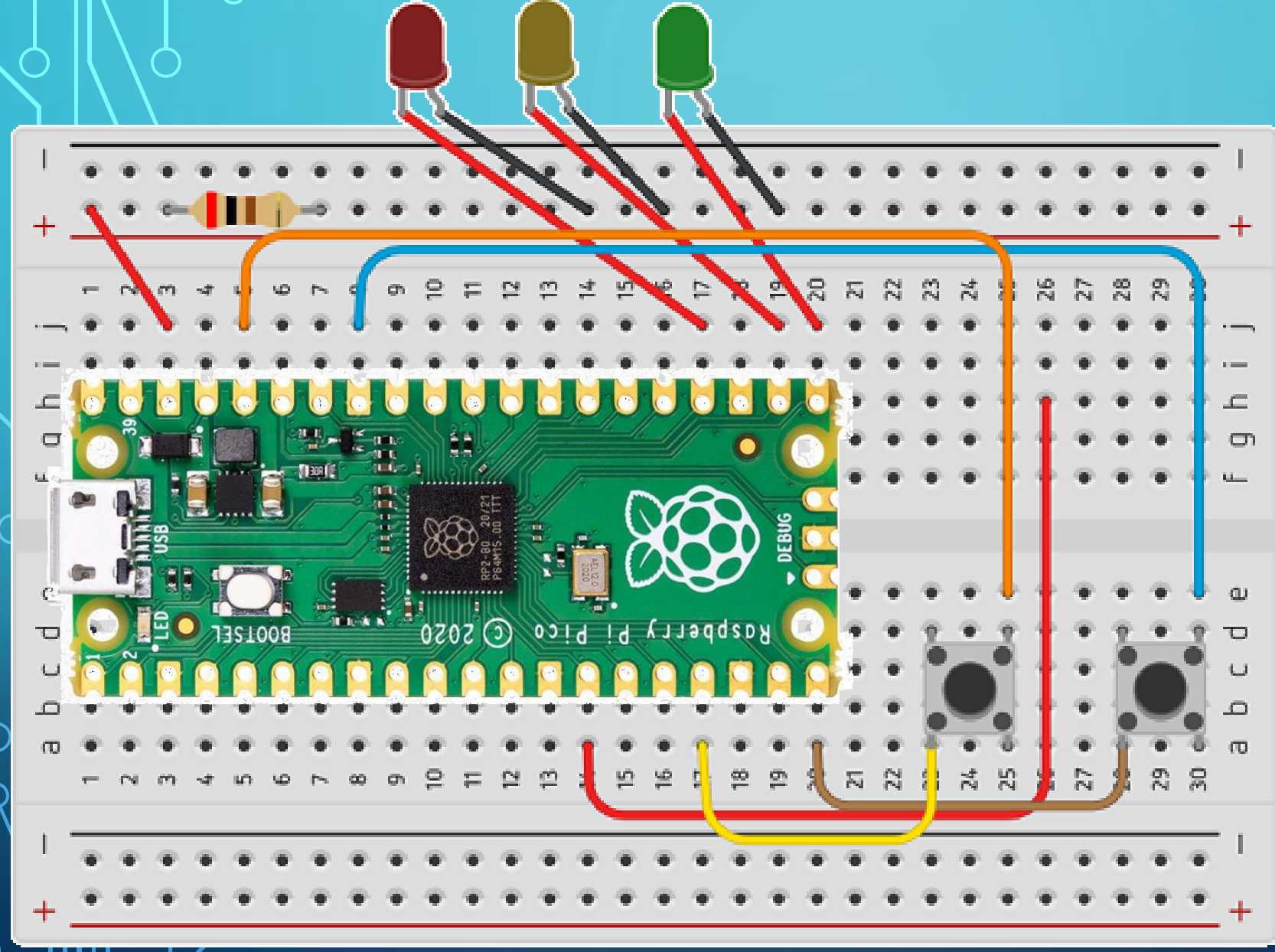
新增第二顆按鈕

```
button_1 = Pin(15, Pin.IN, Pin.PULL_UP)
```

#button_1使用上拉電阻,輸入pin號為GP15,另一端接GND

```
button_2 = Pin(13, Pin.IN, Pin.PULL_DOWN)
```

#button_2使用下拉電阻,輸入pin號為GP13,另一端接3V3(OUT)



按鈕練習：

- 練習題一：新增button_2，使用下拉電阻。並嘗試改寫button_01.py，測試功能。
- 練習題二：button_1亮紅燈、button_2亮綠燈、同時按下黃燈閃爍。
- 練習題三：使用button_1與button_2模擬雙切開關

按鈕練習

- 練習題四：button-1每按下一次，**紅**、**綠**燈狀態互換(**紅**燈on時，**綠**燈off)
觀察練習題四功能是否如預期，如何修正。
- 練習題五：修正練習題三-使用**旗標**。

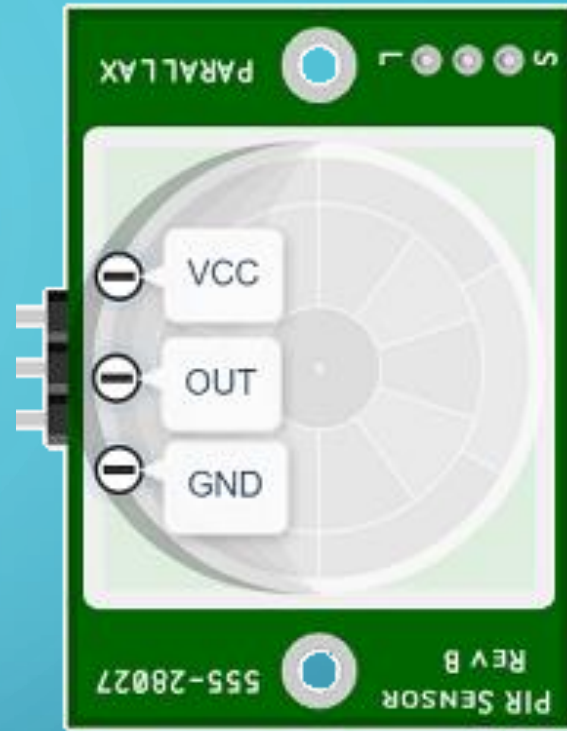
新增人體紅外線偵測器 (PIR HC-SR501)

PIR = Pin(1, Pin.IN, Pin.PULL_DOWN)

#VCC→5V輸出(VBUS)

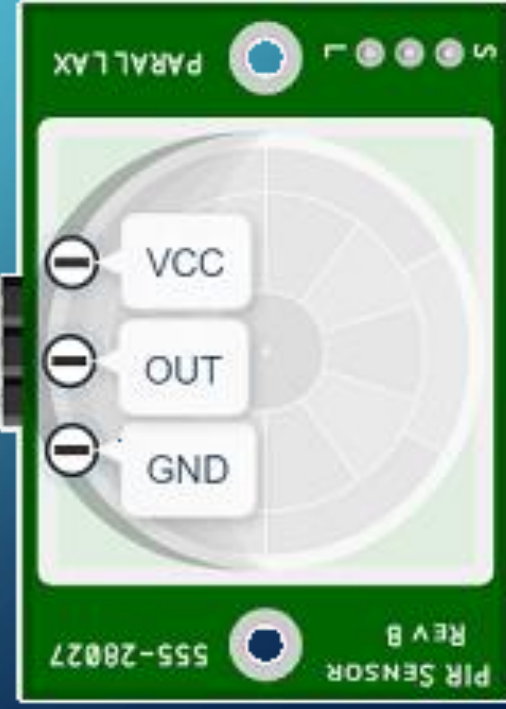
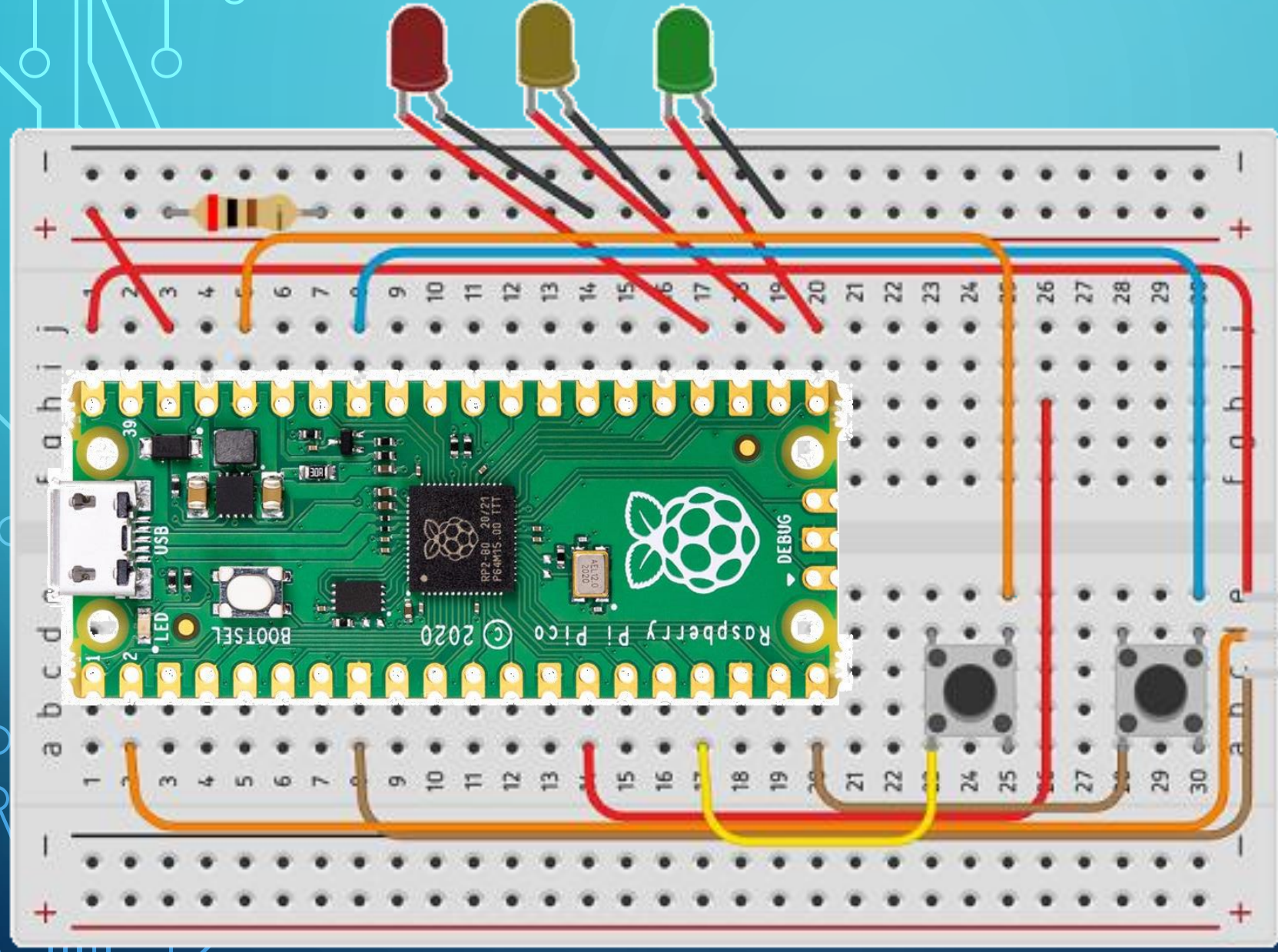
#OUT→GP1

#GND



Finally, you need to connect the power wire. Don't connect this to your Pico's 3V3 pin, though: the HC-SR501 is a 5 V device, meaning that it needs five volts of electricity in order to work. If you wire the sensor to your Pico's 3V3 pin, it won't work – the pin simply doesn't provide enough power.

To give your sensor the 5 V power it needs, wire it to the very top-right pin of your Pico – VBUS. This pin is connected to the micro USB port on your Pico, and taps into the USB 5 V power line before it's converted to 3.3 V to run your Pico's microprocessor. All three HC-SR501 pins should now be wired to your Pico: ground, signal, and power.

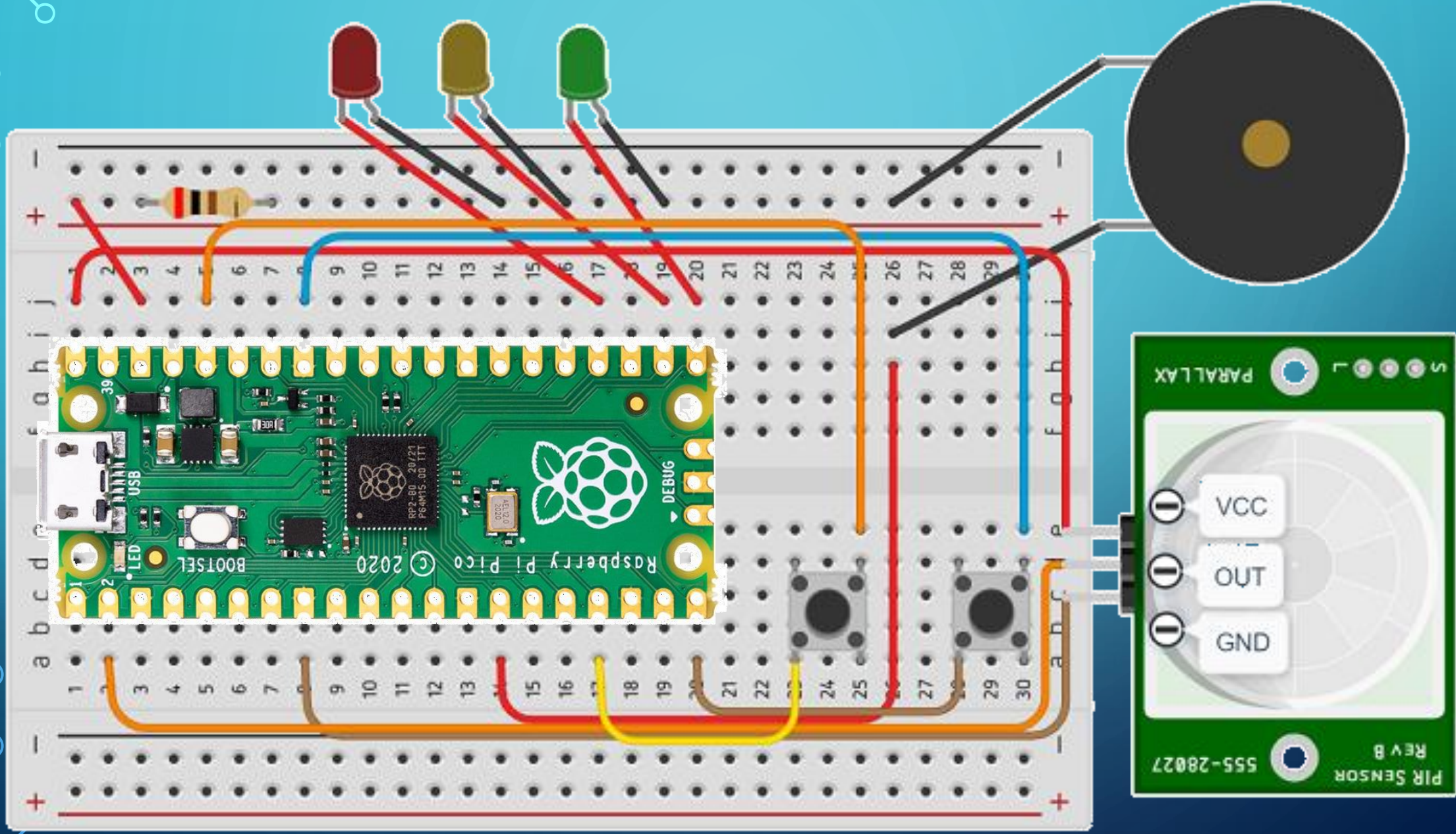


人體紅外線範例

```
#PIR使用範例：PIR_01.py  
from machine import Pin  
from utime import sleep
```

```
PIR = Pin(1, Pin.IN, Pin.PULL_DOWN)
```

```
i=0  
while True:  
    print(i,PIR.value())  
    sleep(1)  
    i+=1
```

新增蜂鳴器

#蜂鳴器使用範例 : buzzer_01.py

from machine import Pin

from utime import sleep

buzzer = Pin(10, Pin.OUT)

For i in range(3):

 buzzer.value(1)

 sleep(0.5)

 buzzer.value(0)

 sleep(0.5)

行人優先權的紅外線感測



行人優先權的紅外線感測-1

```
#pedestrianfirst_IF.py
#使用IF判斷式
def trafficLight():
    ...
def pedestrian_first():
    ...
while True:
    trafficLight()
    if PIR.value() == 1:
        pedestrianfirst()
```

行人優先權的紅外線感測-2：IRQ_01.PY

```
#pedestrianfirst_IRQ.py
#使用中斷 IRQ
#設定當PIR偵測到人體時(電壓 RISING),啟動中斷 IRQ
def trafficLight():
    ...
def IRQ():
    ...
while True:
    PIR.irq(trigger=Pin.IRQ_RISING,handler=IRQ)
    trafficLight()
```

行人優先權的紅外線感測-3 : THREAD_01.PY

```
#pedestrianfirst_thread.py
#在另一個Thread執行偵測PIR狀態的程式
Import _thread
def trafficLight():
def pedestrian_first():
def PIR_sensor_thread():
_thread.start_new_thread(PIR_sensor_thread, ())
while True:
    trafficLight()
    if PIR_sensor == True:
        pedestrian_first()
        PIR_sensor = False
```


TrafficLight()修改

規則1：

逐行點燈
、熄燈

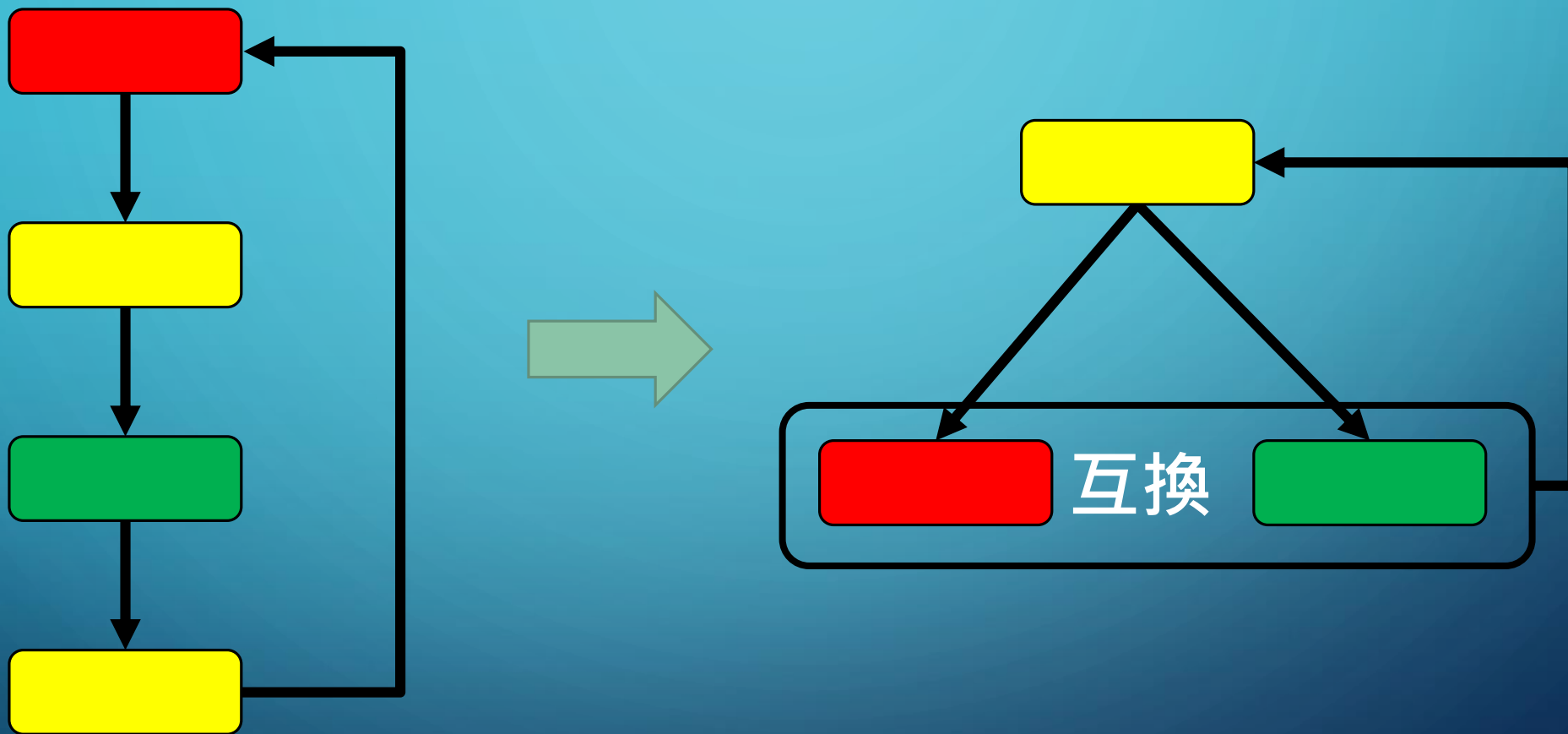
規則2：

先亮黃燈,再
亮紅(綠)燈

規則3：

同一時間只亮
一燈，使用變
數控制

TrafficLight()修改 : `TRAFFICLIGHT2.PY`



TrafficLight()修改 : TRAFFICLIGHT3.PY

led_red.value(**1 == True == 有值**)

	i=0	i=1	i=2
i %3	0	1	2
(i+1)%3	1	2	0
(i+2)%3	2	0	1

	i=0	i=1	i=2
i %3	false	true	true
(i+1)%3	true	true	false
(i+2)%3	true	false	true

	i=0	i=1	i=2
~i %3	true	false	false
~(i+1)%3	false	false	True
~(i+2)%3	false	true	false