

教具名稱	馬達與感測器教具
課程名稱	紅綠燈系統
運算思維	演算法步驟
編撰教師	邱昭士
編撰基地	新北市碧華國小衛星基地
課程影片(有/無)	無

# 大綱

1. 情境主題及目的
2. 情境分析
3. 副程式「紅燈狀態」設計
4. 副程式「綠燈狀態」設計
5. 副程式「小紅人行走」設計
6. 本系統演算法步驟及積木程式堆疊

# 情境主題及目的

**(1) 情境主題：**紅綠燈系統

**(2) 情境目的：**結合搖桿、無源蜂鳴器、RGB LED模組、8\*8點矩陣、伺服馬達的運轉，利用搖桿按鈕，控制紅綠燈狀況。

1. 程式開始執行時，『紅燈狀態』用RGB LED 顯示紅燈，柵欄放下，用8x8LED點矩陣 模擬小紅人立正站立。
2. 當搖桿的按鈕壓下，紅燈變綠燈狀態，RGB LED 變綠燈，開啟柵欄，同時用蜂鳴器播放鳥鳴聲或音樂。
3. 小紅人開始行走持續30秒，倒數15秒，小紅人隨倒數時間越少行走速度越快。
4. 倒數計時結束，變紅燈狀態，小紅人變回立正站立，音樂停止，柵欄放下。
5. 程式等待下一個搖桿按鈕壓下的指令

# 情境分析

## (3) 情境分析：

1. 程式開始執行時，設定『紅燈狀態』：
  - ▶ 用RGB LED 顯示紅燈：數位腳位 9 設為「高」電位、數位腳位 10 設為「低」電位、數位腳位 11 設為「低」電位。
  - ▶ 放下柵欄：伺服馬達 腳位 6 角度 0 度
  - ▶ 設8x8LED陣列模擬小紅人立正站立
2. 當搖桿的按扭壓下，紅燈變綠燈狀態
  - ▶ RGB LED 變綠燈：數位腳位 9 設為「低」電位、數位腳位 10 設為「高」電位、數位腳位 11 設為「低」電位。
  - ▶ 開啟柵欄：伺服馬達 腳位 6 角度 90 度
  - ▶ 用蜂鳴器播放鳥鳴聲或音樂：數位腳位 8 設為「高」電位，腳位 8 播放音調
  - ▶ 同時，小紅人開始行走持續30秒：設 8x8LED陣列 模擬小紅人行走30秒
  - ▶ 小紅人行走計時倒數15秒，小紅人隨倒數時間越少行走速度越快。
  - ▶ 倒數計時結束，變回紅燈狀態
3. 程式等待下一個搖桿按鈕壓下的指令

# 演算法步驟

## (4) 副程式「紅燈狀態」演算法步驟：

01	<b>用RGB LED 亮紅燈：</b> 數位腳位 <b>9</b> 設為「 <b>高</b> 」電位 數位腳位 <b>10</b> 設為「 <b>低</b> 」電位 數位腳位 <b>11</b> 設為「 <b>低</b> 」電位。
02	<b>放下柵欄：</b> <b>伺服馬達</b> 腳位 <b>6</b> 角度 <b>0</b> 度 等待 <b>1</b> 秒
03	<b>8x8 LED陣列：</b> 模擬小紅人立正站立

# 演算法步驟 vs 積木程式堆疊

## (4-1) 副程式「亮紅燈」演算法步驟：

- 01 用**RGB LED** 亮紅燈：  
數位腳位 **9** 設為「**高**」電位  
數位腳位 **10** 設為「**低**」電位  
數位腳位 **11** 設為「**低**」電位。



# 演算法步驟 vs 積木程式堆疊

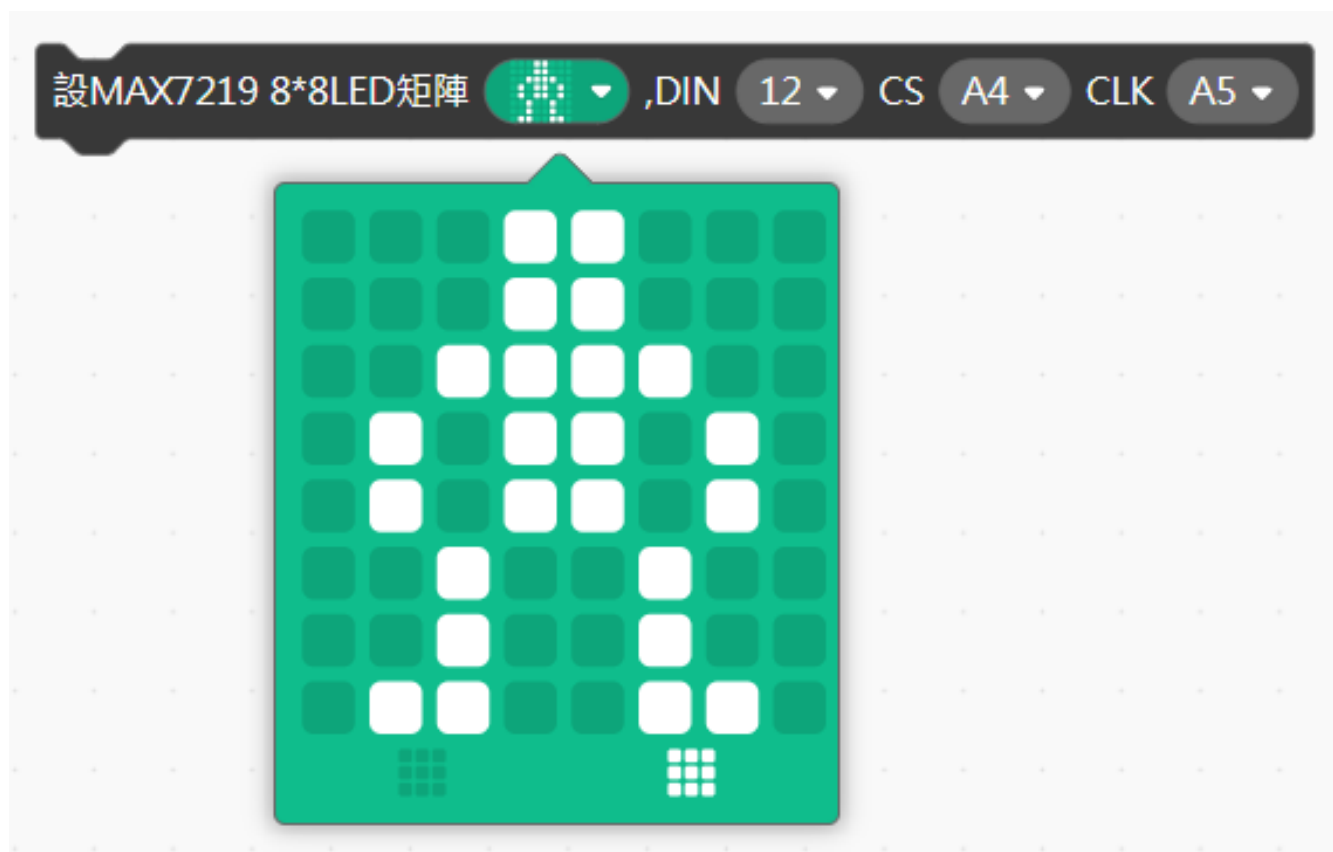
## (4-2) 副程式「放下柵欄」演算法步驟：

02	放下柵欄： 伺服馬達 腳位 6 角度 0 度 等待 1 秒
----	-------------------------------------



# 演算法步驟 vs 積木程式堆疊

03 **8x8 LED陣列**：  
模擬小紅人立正站立





# 演算法步驟 vs 積木程式堆疊

## (4-3) 副程式「紅燈狀態」演算法步驟：

01	副程式「亮紅燈」
02	副程式「放下柵欄」
03	模擬小紅人立正站立

The image shows a Scratch-style block diagram for a simulation. It features three stacked blocks: a 'Define' block for 'Red Light State', a 'Turn on red light' block, and a 'Lower barrier' block. Below these is a dark grey control bar for a MAX7219 8\*8 LED matrix. The control bar includes a green 'Matrix' block with a grid icon, followed by pins for DIN (12), CS (A4), and CLK (A5).

# 演算法步驟

## (5) 副程式「綠燈狀態」演算法步驟：

01	<b>用RGB LED 亮綠燈：</b> 數位腳位 9 設為「低」電位 數位腳位 10 設為「高」電位 數位腳位 11 設為「低」電位。
02	<b>打開柵欄：</b> <b>伺服馬達</b> 腳位 6 角度 90 度 等待 1 秒
03	<b>蜂鳴器播放音調或音樂：</b> 數位腳位 8 設為「高」電位， 蜂鳴器在腳位 8 播放音調.....直到播完 數位腳位 8 設為「高」電位

# 演算法步驟 vs 積木程式堆疊

## (5-1) 副程式「亮綠燈」演算法步驟：

- 01 用**RGB LED** 亮綠燈：  
數位腳位 **9** 設為「**低**」電位  
數位腳位 **10** 設為「**高**」電位  
數位腳位 **11** 設為「**低**」電位。



# 演算法步驟 vs 積木程式堆疊

## (5-2) 副程式「打開柵欄」演算法步驟：

02	打開柵欄： 伺服馬達 腳位 6 角度 90 度 等待 1 秒
----	--------------------------------------



# 演算法步驟 vs 積木程式堆疊

## (5-3) 副程式「播放音樂」演算法步驟：

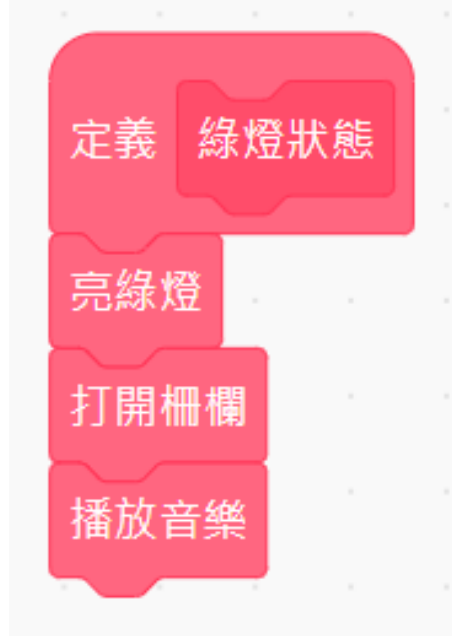
03 **蜂鳴器**播放音調或音樂：  
數位腳位 8 設為「高」電位，  
蜂鳴器在腳位 8 播放音調.....直到播完  
數位腳位 8 設為「高」電位

The image shows a Scratch code editor with three code blocks for playing a tone on a buzzer. The first block is a 'Define' block (highlighted in red) with the name '播放音樂'. The second block is a 'Set digital pin' block with '8' selected for the pin and 'High (1)' selected for the output. The third block is a 'Buzzer plays tone' block with '8' selected for the pin, 'Do, 262' selected for the frequency, and '500 ms' selected for the duration, followed by the text '直到播完' (until finished playing).

# 演算法步驟 vs 積木程式堆疊

## (5-4) 副程式「綠燈狀態」演算法步驟：

01	副程式「亮綠燈」
02	副程式「打開柵欄」
03	副程式「播放音樂」



# 演算法步驟

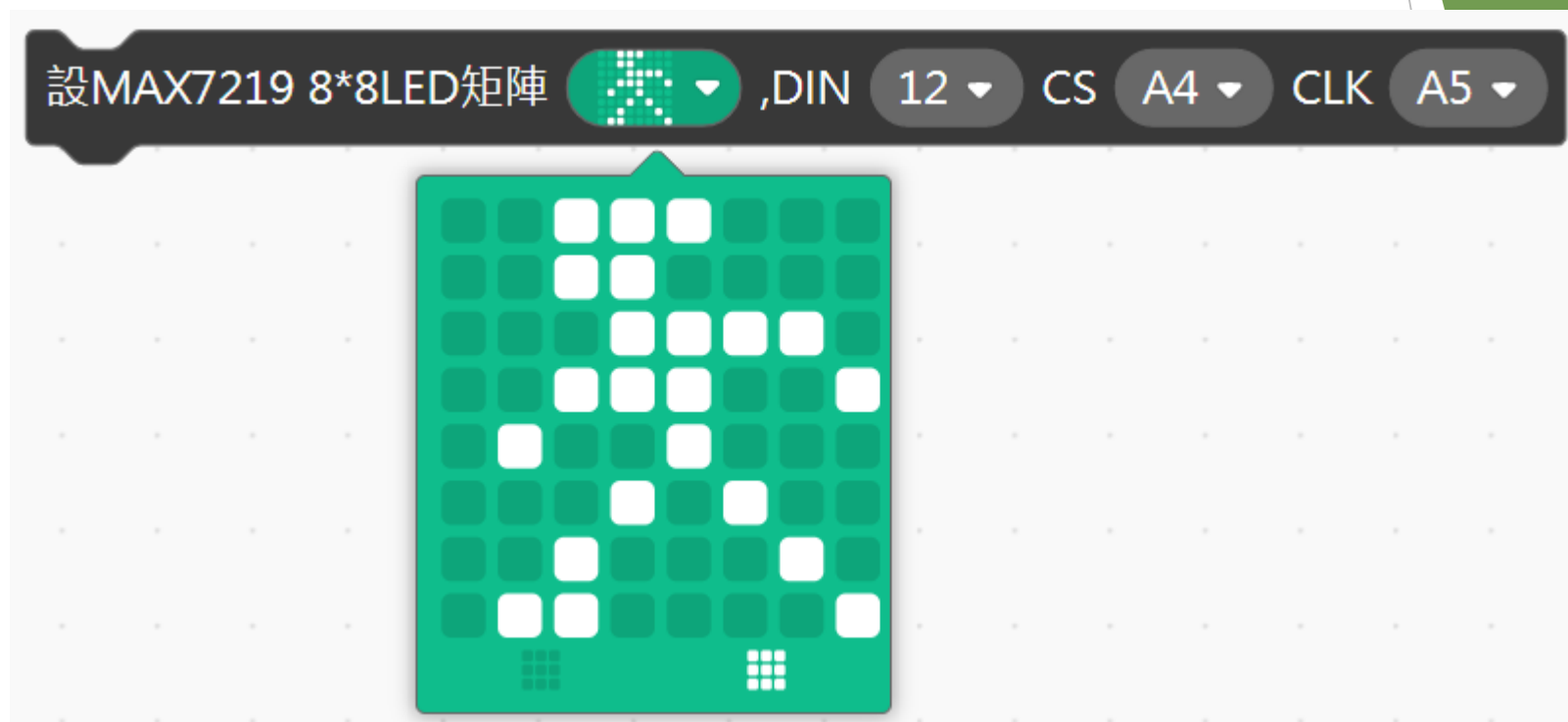
## (6) 副程式「小紅人行走」演算法步驟：

模擬「小紅人行走 30 秒，倒數15秒，越走越快」

01	重複迴圈10次
02	模擬小紅人立正站立，等待0.75秒
03	模擬小紅人往前走，等待0.75秒
04	重複迴圈10次
05	模擬小紅人立正站立，等待0.5秒
06	模擬小紅人往前走，等待0.5秒
07	重複迴圈10次
08	模擬小紅人立正站立，等待0.25秒
09	模擬小紅人往前走，等待0.25秒

# 演算法步驟 vs 積木程式堆疊

(6-1) 模擬小紅人往前走：

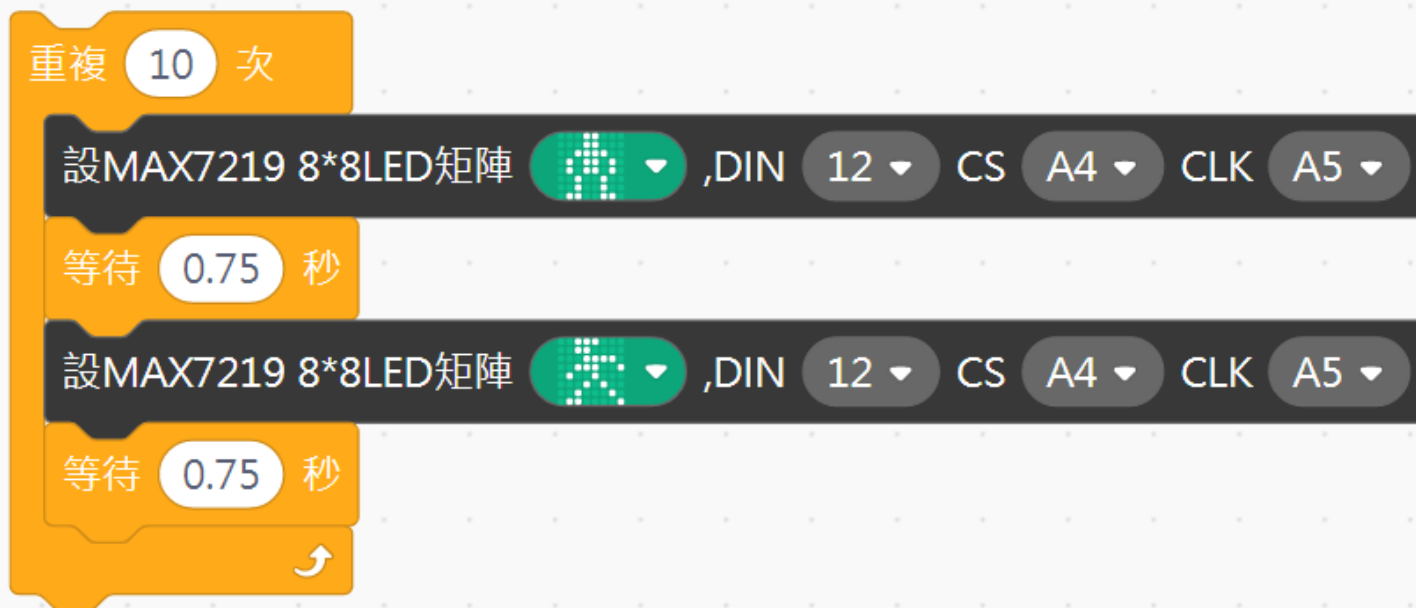




# 演算法步驟 vs 積木程式堆疊

## (6-2) 副程式「小紅人行走」演算法步驟1-3：

01	重複迴圈10次
02	模擬小紅人立正站立，等待0.75秒
03	模擬小紅人往前走，等待0.75秒



# 演算法步驟 vs 積木程式堆疊

## (6-3) 副程式「小紅人行走」演算法步驟4-6：

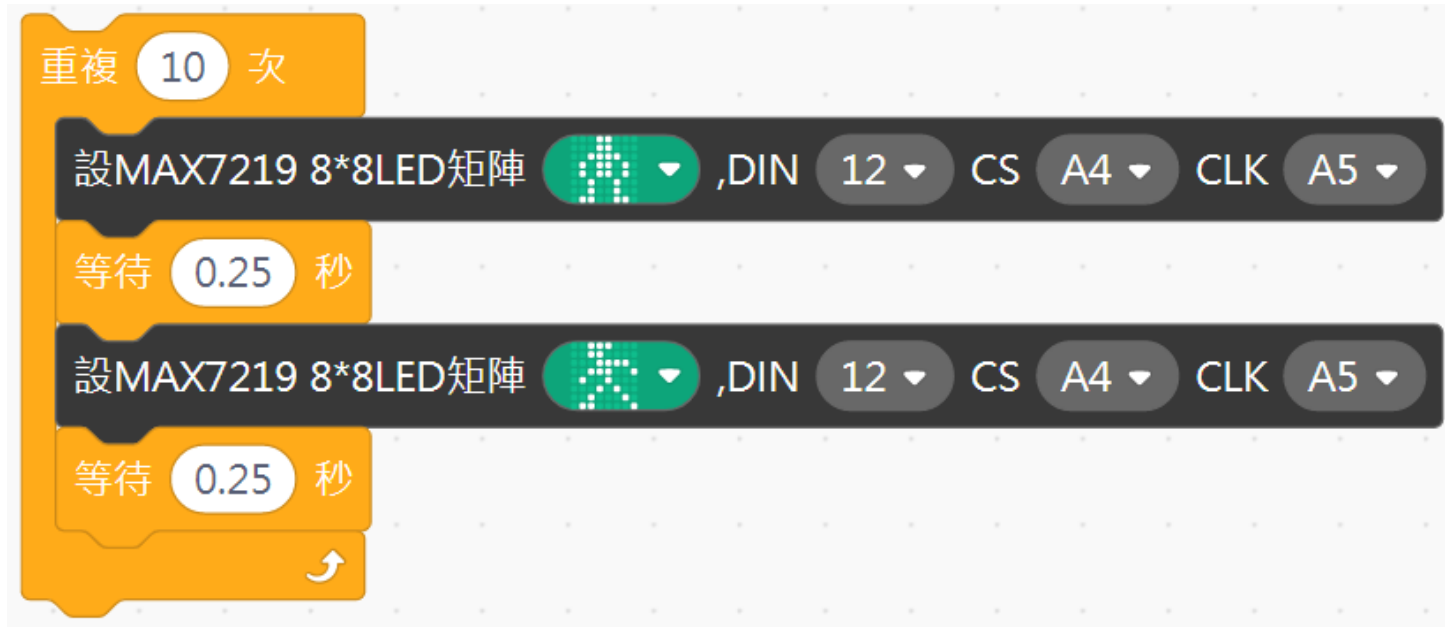
04	重複迴圈10次
05	模擬小紅人立正站立，等待0.5秒
06	模擬小紅人往前走，等待0.5秒

The image shows a Scratch-style block diagram for a loop. It starts with an orange 'Repeat' block set to 10 times. Inside the loop, there are two identical sequences of blocks: a dark grey block for 'Set MAX7219 8\*8LED matrix' with pins DIN: 12, CS: A4, CLK: A5, followed by an orange 'Wait 0.5 seconds' block. The loop ends with a small arrow icon.

# 演算法步驟 vs 積木程式堆疊

## (6-4) 副程式「小紅人行走」演算法步驟7-9：

07	重複迴圈10次
08	模擬小紅人立正站立，等待0.25秒
09	模擬小紅人往前走，等待0.25秒

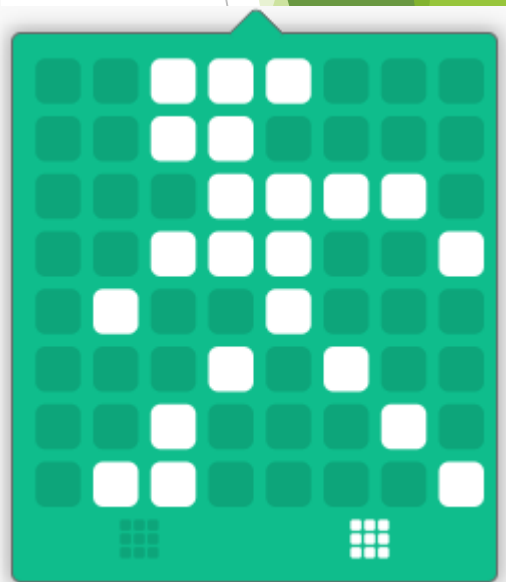
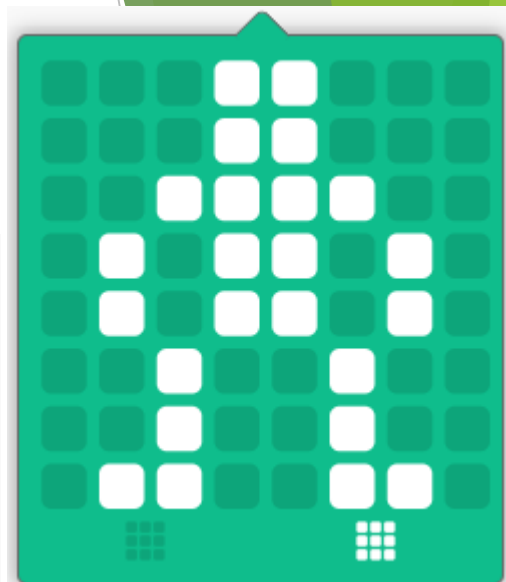


# 積木程式堆疊

## (6-5) 副程式「小紅人行走」

The diagram shows a Scratch-style block diagram for a sub-program named "小紅人行走" (Little Red Man Walking). The blocks are as follows:

- 定義 小紅人行走** (Define Little Red Man Walking)
- 變數 秒數 設為 0.75** (Variable seconds set to 0.75)
- 重複 3 次** (Repeat 3 times)
- 重複 10 次** (Repeat 10 times)
- 設MAX7219 8\*8LED矩陣 [Little Red Man] ,DIN 12 CS A4 CLK A5** (Set MAX7219 8\*8 LED matrix to show Little Red Man)
- 等待 秒數 秒** (Wait seconds)
- 設MAX7219 8\*8LED矩陣 [Little Red Man] ,DIN 12 CS A4 CLK A5** (Set MAX7219 8\*8 LED matrix to show Little Red Man)
- 等待 秒數 秒** (Wait seconds)
- 變數 秒數 改變 -0.25** (Variable seconds change by -0.25)



# 演算法步驟

## (7) 「紅綠燈系統」演算法步驟：

01	副程式： <b>紅燈狀態</b>
02	<b>重複無限次</b> 迴圈開始
03	<b>判斷</b> ：如果搖桿的按鈕被壓下
04	<b>成立</b> ：副程式： <b>綠燈狀態</b>
05	副程式： <b>小紅人行走</b>
06	副程式： <b>紅燈狀態</b>
	重複迴圈結束

# 積木程式堆疊

## (7-1) 紅綠燈系統

當 被點擊

紅燈狀態

重複無限次

如果 讀取數位腳位 7 INPUT\_PULLUP 註 = 1 那麼

綠燈狀態

小紅人行走

紅燈狀態