STEM_6

自　學　教　材

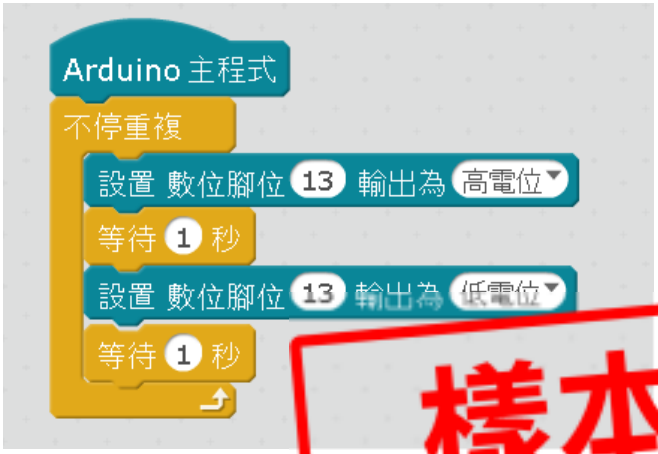
專案一：搶答機

目錄

連接到 Arduino 主板的 GND 針腳。

連接到負極的電阻用於防止過大的電流損壞 LED，因此它被稱為限流電阻。所有 LED 都必須配有限流電阻。

在成功構建電路之後，讓我們打開 mBlock 軟件並編寫我們的第一個 Arduino 程式。



在上面的程式中，我們將輸出設置為「高電位」以點亮 LED，然後我們將輸出設置為「低電位」以熄滅 LED。我們在每個「設置數位腳位 X 輸出為 X」積木之後放置一個「等待 X 秒」積木來實現閃爍效果。

基本上，LED 會點亮一秒鐘，然後熄滅一秒鐘。因為我們把程式放在一個「不停重複」積木中，所以程式將永遠運行，直到我們切斷 Arduino 主板的電源為止。

[資源檔案：Controlling-LEDs.sb2]

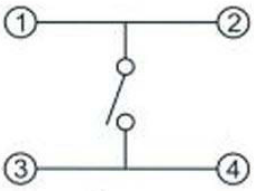
4. 作業 1

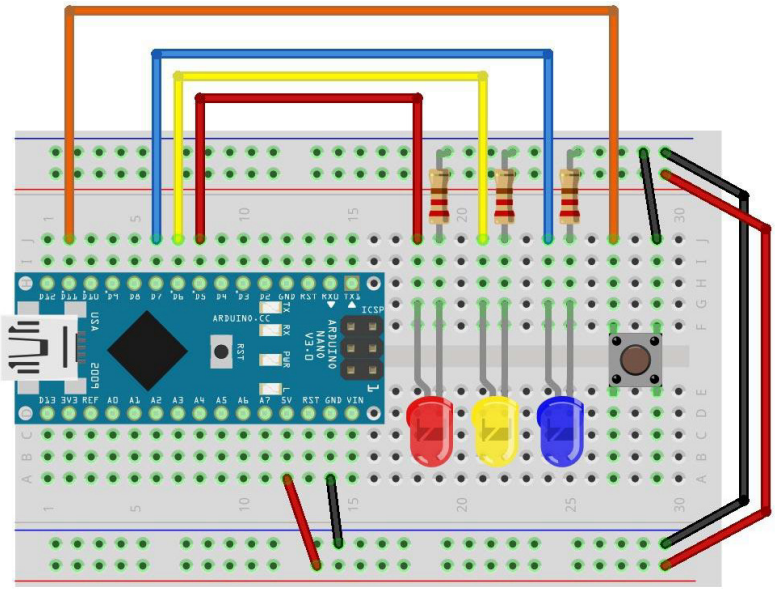
1. 將 LED 的連接針腳從數位針腳 13 更改為 Arduino 主板的數位針腳 11。(提示：你需要修改電路以及程式)
2. 修改上述程式，使閃爍速度增加一倍。
3. 在數位針腳 5 添加一個 LED，然後令兩個 LED 一起閃爍。

5. 使用按鈕

開關 (switch) 是可以閉合和斷開電路的電子零件。有兩種主要的開關類型，撥動開關和瞬時開關。

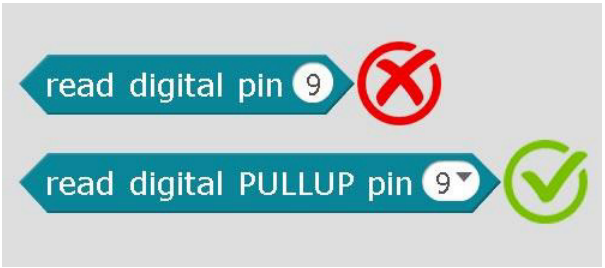
輕觸按鈕 (tactile button) 是僅在按下按鈕時導接通的電子開關。典型的輕觸按鈕有 4 個針腳，針腳分為兩組，即使未按下按鈕，每組中的針腳也會接通。當按下按鈕時，所有 4 個針腳都接通。輕觸按鈕屬於瞬時開關。





根據上圖構建電路。將紅色 LED 連接到 Arduino NANO 主板的數位針腳 5。將黃色 LED 連接到數位針腳 6。將藍色 LED 連接到數位針腳 7。不要忘記替每個 LED 加上限流電阻。

把輕觸按鈕的其中一個針腳連接到 Arduino 主板的數位針腳 11，另一組的其中一個針腳接地。請注意，這個連接方法不需要用到電阻。這是因為我們使用了 Arduino 主板的一個特有功能──內置上拉電阻（built-in pullup resistor）。



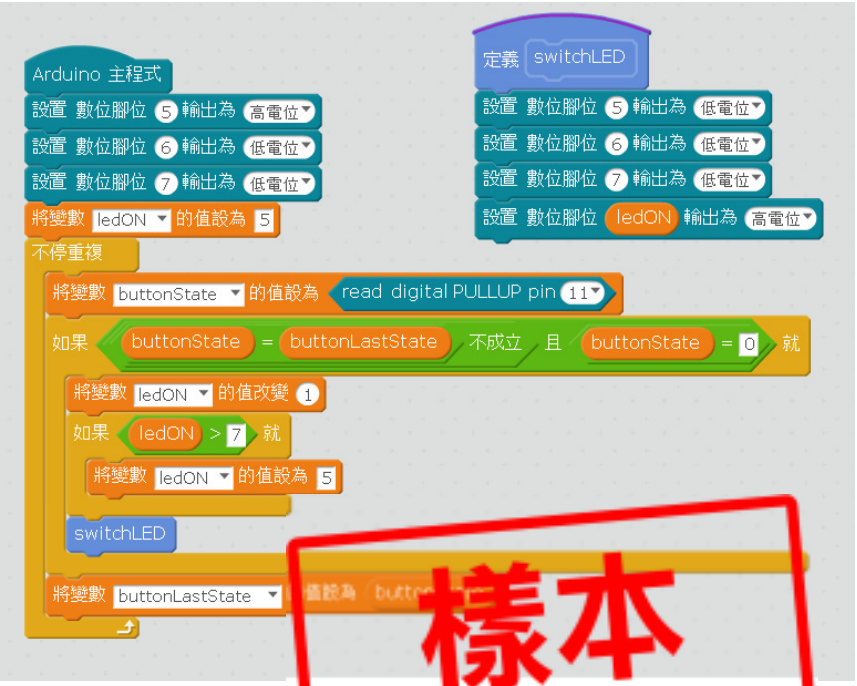
要啟用內置上拉電阻，我們必須使用「read digital PULLUP pin X」積木。而不是原來的「數位腳位 X」積木。

要以編程方式使用輕觸按鈕，我們會應用一種稱為「信號邊緣檢測」(signal edge detection)或「狀態變化檢測」(state change detection) 的技術。我們可以使用這種技術來檢測按鈕何時被「按下」以及何時被「釋放」。

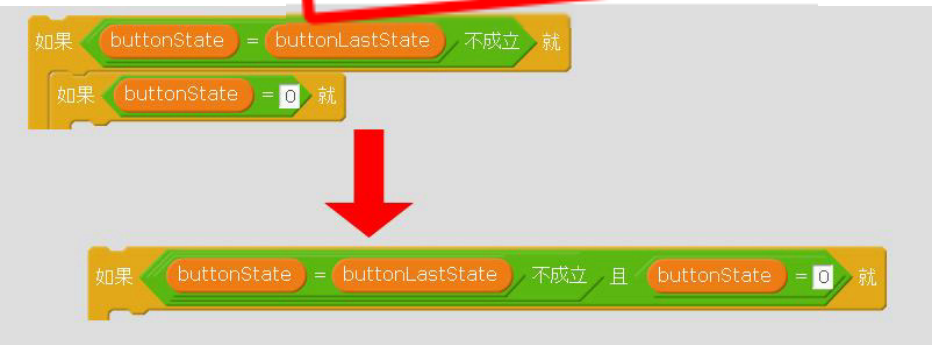
在 Arduino 編程環境中，按鈕可以有兩種狀態。按下按鈕時，「read digital PULLUP pin X」積木將傳回數值 0。當未按下按鈕時，「read digital PULLUP pin X」積木將傳回數值 1。

我們使用兩個變數 buttonState 和 buttonLastState 來追踪每個循環裡面，當前的按鈕狀態和對上一個按鈕狀態。除非按鈕狀態有所改變，即

buttonState 不等於 buttonLastState，否則程式將不會執行任何動作。每當使用者按一下按鈕時，都會包含按下和釋放兩個動作。作為程式員，我們可以決定將我們的程式碼放在按下時執行，還是釋放時執行。



在上面的程式中，當使用者每次按下按鈕時，不同的 LED 將會亮起。



為了簡化程式，我們將兩個「如果..就..」積木用一個「..和..」積木結合在一起。

STEM_6

Self-learning Guide

PROJECT 2 LUCKY WHEEL

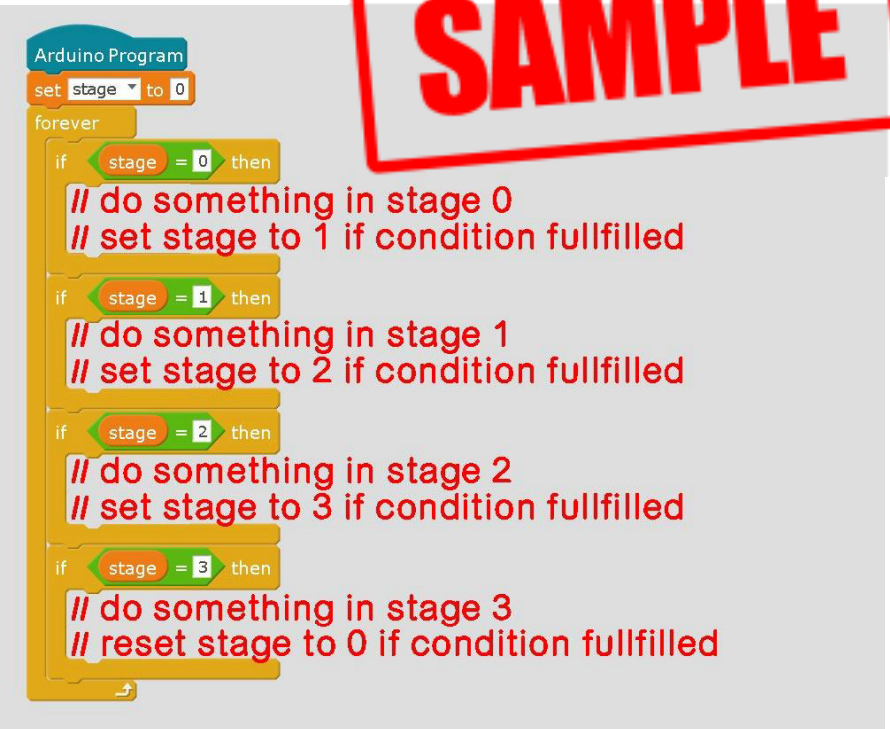
Table of Contents

9. Game Structure

In this section, we will be talking purely about programming. More precisely, we will be talking about the programming structure of a game.

In general, for all games, there are different **stages** or **levels**. The game would not proceed to next stage / level after finishing the previous one. The game would restart at the beginning after the game is over.

To mimic the game stages, we would use a variable called **stage**. Initially, we set the variable **stage** to **0**. If one stage is completed, we will advance the variable stage by **1**. If the whole game is completed, the variable **stage** would be reset to **0**.

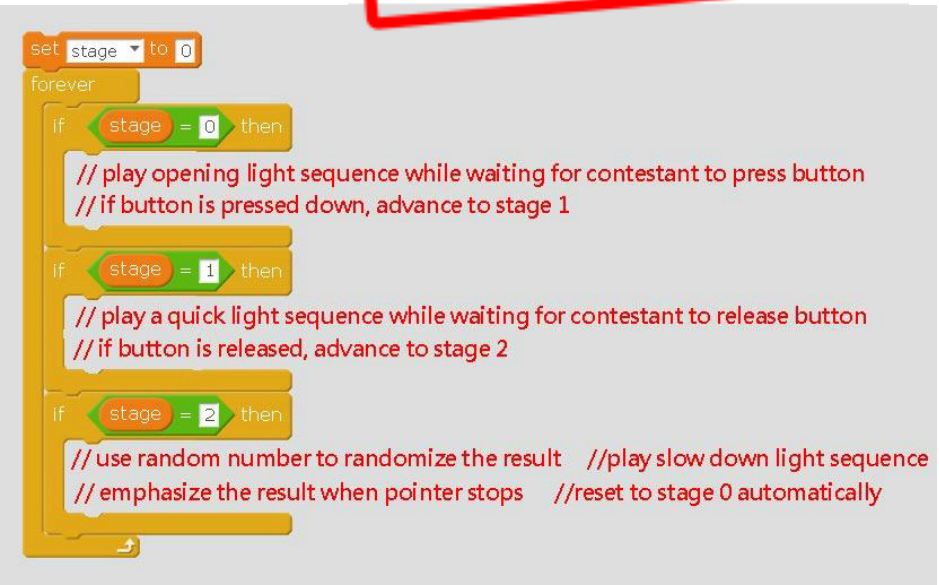


In the above example, the whole program is controlled by a single variable **stage** and a series of **if then** blocks. Because the variable **stage** can only hold one value at a time, the **if then** blocks would prevent programming codes of other stages from running.

You can design what happen in each stage. You can light up different LEDs at different stage. You can play different sound at different stage. The only limitation is your imagination.

You can also program a condition at each stage for the game to advance to the next stage. The condition can be anything programmable, from as simple as a single button click, or just let the program to wait some time, to as complex as asking the user to perform a series of tasks.

There can also be multiple conditions at each stage, one condition leads to the completion of that stage, while the other condition causes the game end.

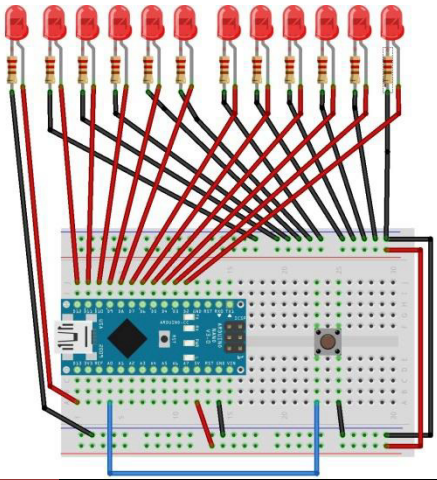


Let us spend a minute to think of the different stages of a lucky wheel system. At stage 0, the lucky wheel would be waiting for the contestant to press the button. When the button is pressed, our program would advance to stage 1. In our program, the wheel would not slow down until the button is release. So in stage 1, it would play a quick light sequence to mimic the speed up of the wheel. When the button is released, the program would advance to stage 2. In stage 2, the wheel would start to slow down and eventually the pointer would stop at one position.

To make our lucky wheel better, our program would play an opening light sequence while waiting in stage 0. And after the pointer stops, we would blink the LED of the winning position for several times to emphasize the result. After that, our system would reset to stage 0 and wait for another button press.

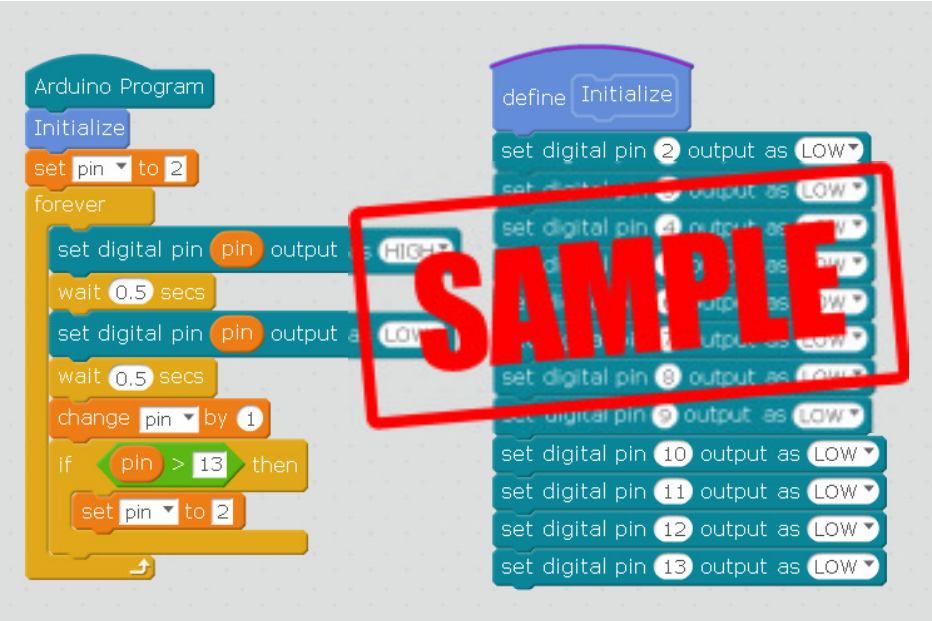
10. LED light sequences

We will be using many light sequences in our lucky wheel program, so how to program different light sequences? Let us built a circuit with 12 LEDs according to the below diagram.



There are a total of twelve LEDs connecting to digital pin 2 – 13 of the Arduino board. There is a push button connecting to analog pin A0 of the Arduino board. The analog pin A0 is also called digital pin 14.

There are two essential techniques in programming a light sequences. Using variables to represent output pin number and calculating how long does it take for each LED to turn on and off.



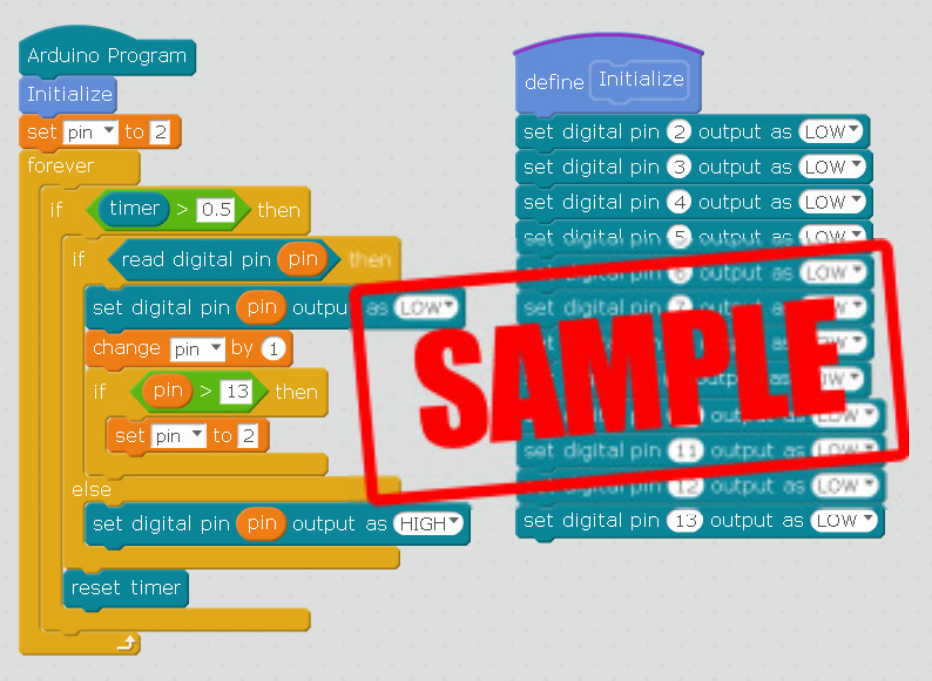
[light-sequence-01.sb2]

In the above program, a variable pin is used to determine which LED to turn on and turn off. And we change the value of pin in each loop from 2 to 13. When the value of pin is bigger than 13, we reset it back to 2.

Two wait blocks are used to control the time duration for each step. In this simple example, the time duration is between each step is fixed.

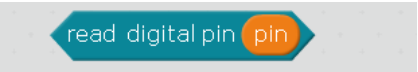
But the above example has a big problem. The **wait** blocks would affect other normal functions such as button state change detection mentioned in section 5. This means we cannot detect button press and button release in the above program.

The solution is simple, ditch the wait blocks and replace them with timer blocks.



[light-sequence-02.sb2]

The second program does the same thing as the first program. Except for using the timer block to control the time duration, there is one more special technique worth mentioning. We use the **read digital pin X** block to check if the LED is ON or OFF.



In the second program, inside each loop, if the target LED is OFF, we turn it on. If the

