

PROJECT 1 TRAFFICE LIGHT SIMULATION

Table of Contents

1. Hardware List	2
2. Traffic Light	2
3. LED and Electric Circuit	3
4. Blinking an LED	5
5. Simulating a Traffic Light	6
6. Using Timer Block.....	8
7. Assignment 1.....	9
8. Controlling the brightness of LED	10
9. Assignment 2.....	13
10. Using momentary switch as input	13
11. Using momentary switch as toggle switch.....	15
12. Switch Debouncing.....	18
13. Assignment 3.....	20
14. Extended Activity	20
15. Further Readings.....	20

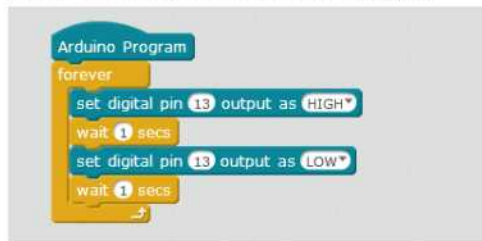
*** If you omit the current limiting resistor in the above circuit, the LED probably would not burn out immediately. It is because the digital output pin of the Arduino UNO board is an output pin of a microcontroller, and the microcontroller would limit the output current to 40mA (according to Arduino specification). A current of 40mA might not destroy the LED at once, but it might shorten the life of the LED significantly. Also, using an LED without current limiting resistor is a bad practice and is not recommended.

4. Blinking an LED

We have now successfully built the circuit, let us open the mBlock application and write a Scratch program to blink the LED.

To switch to Arduino Mode, we can select **Edit > Arduino Mode** in the mBlock application. The Stage area would be hidden, because in Arduino Mode we won't be using the Stage or the Sprites. Instead, an Arduino Programming Language area would appear on the right. Also, some of the unneeded block categories would be disabled in Arduino Mode.

In Arduino Mode, we would start all programs using the **Arduino Program** Block which is located in the **Robots** category.



First, we add the **Arduino Program** block.

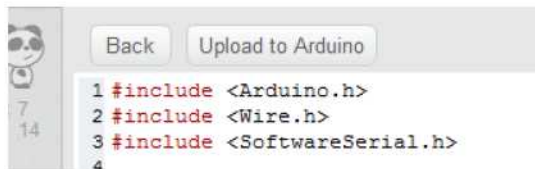
Then, we add a **Forever** block, because we want the LED to blink constantly.

Then, in the **Robots** category, drag a **set digital pin 9 output as HIGH** block inside the forever loop and change the pin number to 13.

Then, add a **wait 1 secs** block.

Then, add another **set digital pin 9 output as HIGH** block. Change the pin number to 13 and the output as LOW

Then, add another **wait 1 secs** block.



```
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4
```

Now, our program is finished. We can click the **Upload to Arduino** button and the LED will be blinking. Don't forget to connect the Arduino UNO board to your computer using the USB cable and select the correct COM port before uploading your program.

In the above program, we need to change the pin number to 13 because our LED is connected to the pin 13 of the Arduino UNO board. When we set the output to HIGH, the pin would output a 5V voltage, which would drive the current through the circuit and turn on the LED light.

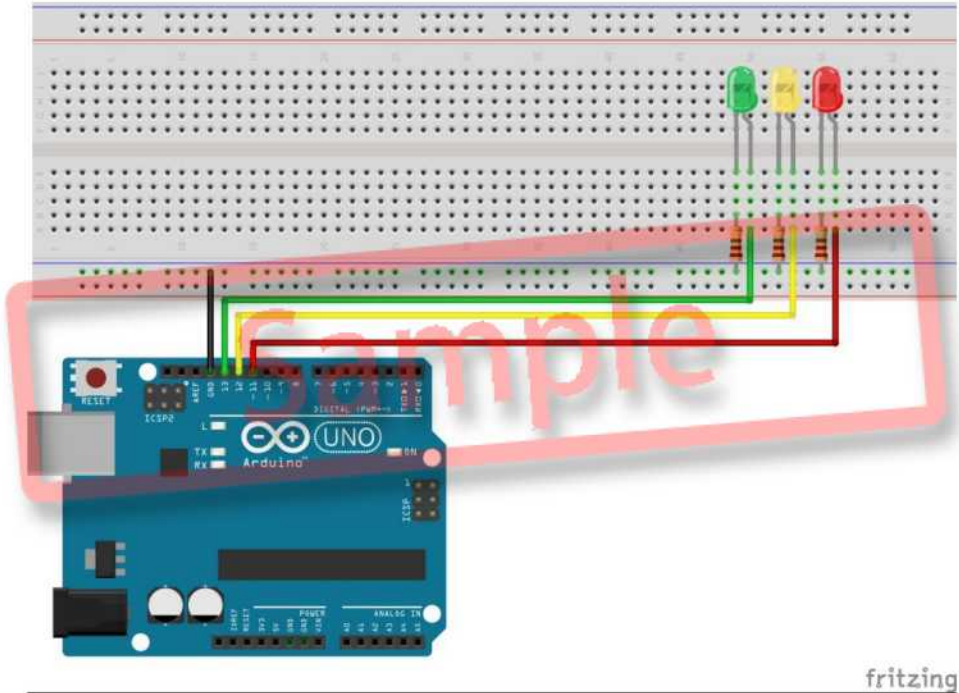
It is a common practice in Arduino programming to put our main program in a **forever** block. Because we would want the program to run forever each time the Arduino board starts.

[Resources file: Blinking-an-LED.sb2]

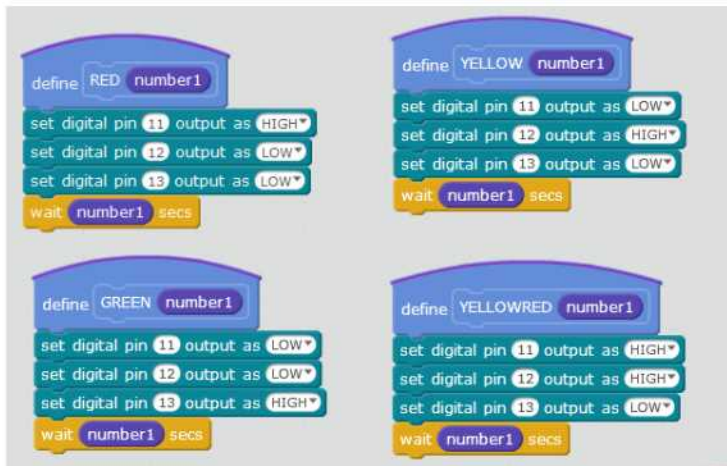
5. Simulating a Traffic Light

Now that we know how to control LEDs with Arduino, let us simulate a single traffic light. Let us assume the green light phase last for 5 seconds and the same for the red

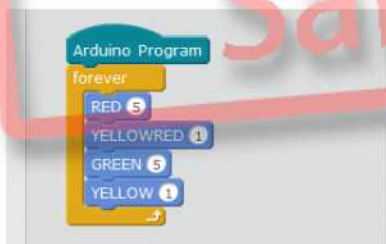
light phase. The duration for yellow phase and red-yellow phase are both one second. First, let us build the circuit as in the below diagram. Let us connect the Green LED to pin 13, Yellow LED to pin 12 and Red LED to pin 11. Do not forget to connect a current limiting resistor to each LED.



Then, let us create four **custom blocks** for each colour phases of the traffic light cycle, and name them **GREEN**, **YELLOW**, **RED** and **YELLOWRED** respectively. In each custom block, we will need a number input in each **custom block** to indicate the phase duration.



Then, let us build the traffic light cycle. Let us start with the RED phase and wait for 5 seconds. Then, YELLOWRED and wait for 1 second. Then, GREEN for 5 seconds and YELLOW for 1 second.



Then let us click the **Upload to Arduino** button, and the three LEDs would start to turn on and off just like a traffic light.

[Resources file: Simulating-a-Traffic-Light.sb2]

6. Using Timer Block

When we need to manipulate time in Arduino programming, we would turn to the **wait** block at first. But we can also use the **timer** block to achieve the same result.

The **timer** block is like a variable which store the internal timer's time. The block starts

Project I Traffic Light Simulator - Assignment 1

1. Add a secondary pedestrian traffic light to the above program (There would be a total of 5 LEDs in the whole system).
 - 1.1 Determine and write down how many phases are needed in this system.
2. Modify the program to fit the below requirements
 - 2.1 Green light duration for car: 20 seconds
 - 2.2 Green light duration for pedestrian: 15 seconds
 - 2.3 Flash the pedestrian green light for the last 5 seconds
 - 2.4 Introduce an ALL RED phase (for 1 seconds) in both directions to clear any traffic remaining in the intersection.

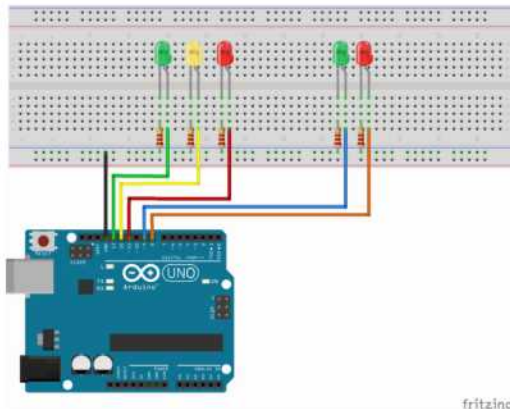
Suggested Answers

Suggested Answers:

Answer 1.1 - The Traffic Light Cycle:

		LED1 Red Car	LED2 Yellow Car	LED3 Green Car	LED4 Red Pedestrian	LED5 Green Pedestrian
Phase1	ALL-RED (1 sec)	ON	OFF	OFF	ON	OFF
Phase2	PEDESTRIAN-GREEN (10 sec)	ON	OFF	OFF	OFF	ON
Phase3	PEDESTRIAN-FLASH (5 sec)	ON	OFF	OFF	OFF	ON (flashing)
Phase4	ALL-RED (1 sec)	ON	OFF	OFF	ON	OFF
Phase5	CAR-YELLOWRED (1 sec)	ON	ON	OFF	ON	OFF
Phase6	CAR-GREEN (20 sec)	OFF	OFF	ON	ON	OFF
Phase7	CAR-YELLOW (1 sec)	OFF	ON	OFF	ON	OFF

Answer 2 - The Circuit on Breadboard:



fritzing

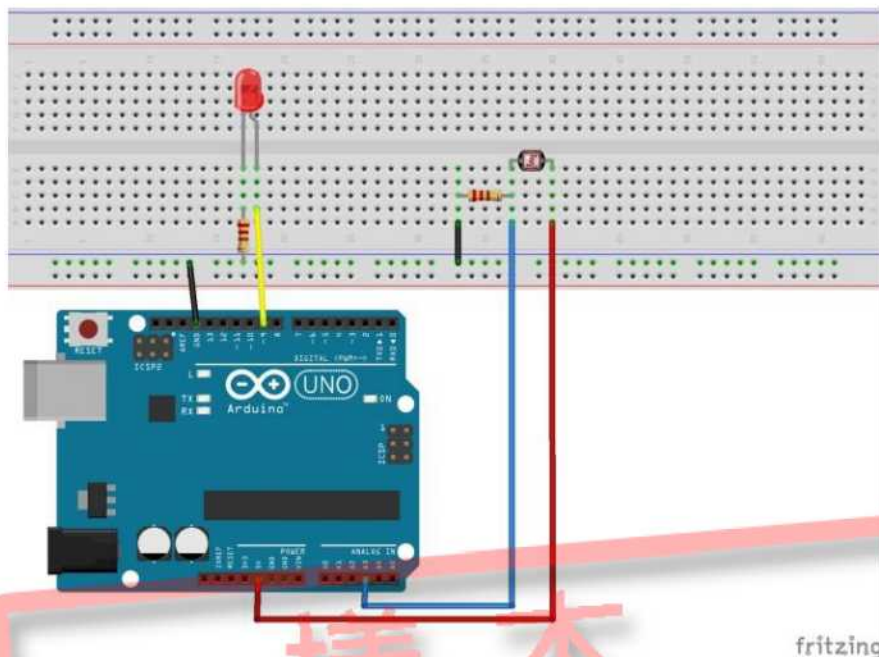
[Resources File: project-1-assignment-1.fzz]

專案二：音樂燈光匯演

目錄

1. 硬件清單	2
2. 蜂鳴器	2
3. 使用壓電式蜂鳴器	3
4. 作業 1	7
5. 光敏電阻	7
6. 使用光敏電阻	7
7. 控制多個 LED	9
8. 作業 2	15
9. 一個簡單的音樂燈光匯演	15
10. 作業 3	18
11. 移位暫存器	18
12. 使用移位暫存器	19
13. 使用獨立電源	26
14. 延伸活動	27
15. 延伸閱讀	27

接到 5V，並將光敏電阻的另一個針腳連接到 Arduino 主板的模擬輸入針腳 (A0 - A5)。我們還需要用一個 10K 電阻，將光敏電阻的第二個針腳連接到 GND。



上述接駁光敏電阻的電路稱為「分壓器」(Voltage Divider)。「分壓器」輸出的電壓高低，是由兩個電阻的電阻值的比值來決定。如果省略了接地的電阻，無論光敏電阻的電阻值是多少，模擬輸入針腳都只會讀取到 5V。

Arduino 主板的模擬輸入針腳就像電壓計，它們可以量度 0-5V 的電壓，並且傳回 0 到 1023 之間的值。傳回的值與輸入電壓成正比。

當光的強度低時，光敏電阻的電阻將增加，因此模擬輸入針腳將讀取到較低的電壓。而當光強度增加時，光敏電阻的電阻值將降低，於是模擬輸入針腳將讀取得到較高的電壓。

讓我們看看應用光敏電阻的程式的例子。



在「不停重複」積木入面的主程式中，我們不停地讀取 A3 針腳傳回的值（我們的光敏電阻就是接駁在 A3），並把此數值儲存到變數 readValue。readValue 的數值會是 0-1023。然後，程式會根據 readValue 的值來閃爍接駁到針腳 9 的 LED。

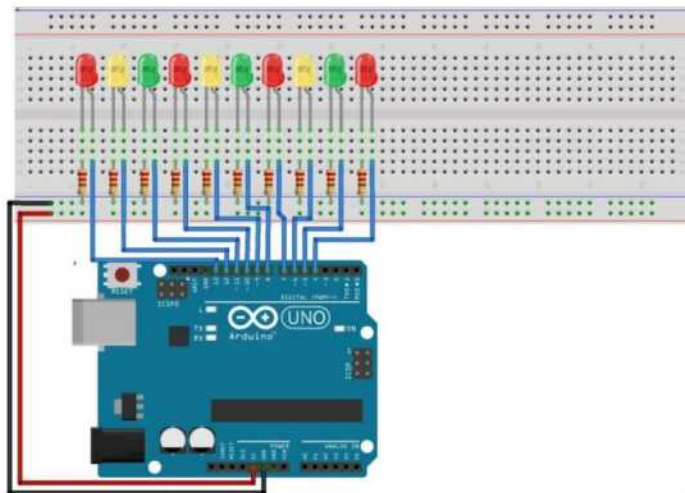
如果 readValue 的值較大，LED 就會閃得比較慢。如果 readValue 的值比較小，LED 就會閃得比較快。

你可以嘗試用不同的東西覆蓋光敏電阻來測試程式。如果用手來覆蓋，會稍微提高閃爍的頻率。而如果使用金屬等不透光的東西來覆蓋，LED 便會閃爍得更快。

[資源檔案：Using-the-LDR-sensor.sb2]

7. 控制多個 LED

在這一章中，我們會使用 10 個 LED 來演示燈光效果，讓我們學習如何控制一堆 LED 吧。請先根據下圖構建電路。10 個 LED 分別連接到 Arduino UNO 主板的數字針腳 4 - 13。不要忘記為每個 LED 加上限流電阻。



fritzing

每個 LED 可以由一個「設置數位腳位」積木來控制。例如，如果我們想要交替亮起 LED，我們可以使用以下程式。

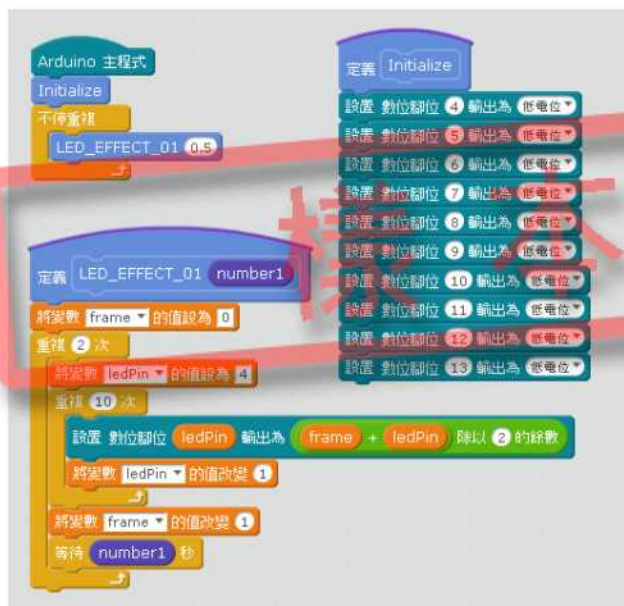


當我們嘗試創建一些 LED 燈光效果時，我們可以想像整個燈光效果是一齣動畫，而每一組 LED 開關組合，就是一個幀 (frame)。例如，在上例的燈光效果中，

我們便有 2 個幀。在第一幀中，只有偶數 LED 會亮起。在第二幀中，只有奇數的 LED 會亮起。程式中的「等待」積木就像「幀速率」(frame rate)。在上例中，我們在每幀後等待 0.5 秒，所以幀速率便是每秒 2 幀 ($1 / 0.5 = 2$)。

[資源檔案：Controlling-Multiple-LEDs-1.sb2]

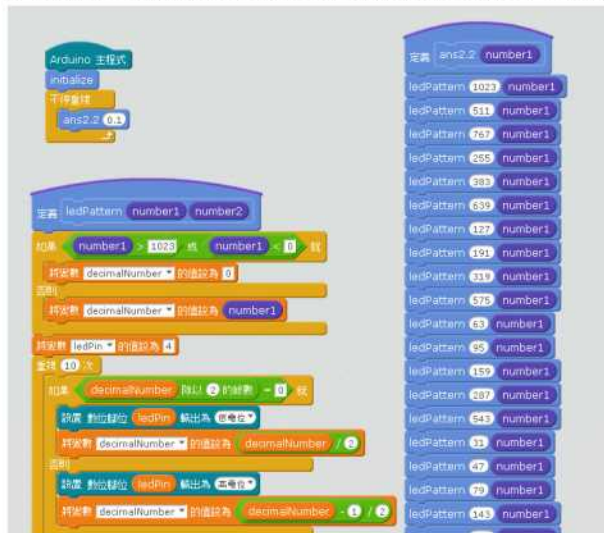
在上面的例子中，我們只有 10 個 LED 和 2 個幀，但程式已經很長了。試想像如果燈光效果有 20-30 個幀，程式便會非常長。所以我們通常會用一些數學方法來縮短程式。試看下列。



要讓 Arduino 主板知道我們將使用針腳 4-13 作為輸出，我們必須在程式啟動時為每個針腳建立一個「設置數位腳位」積木。我們把所有「設置數位腳位」積木都放在自定義積木 Initialize 中。如果沒有初始化數位輸出針腳，程式之後的輸出可能會不正常。

在自定義積木 LED_EFFECT_01 中，我們使用數學方法模擬出相同的 LED 燈光效果。在每個幀中，我們使用「重複 10 次」積木來循環設置 10 個數位針腳。每

2. 簡單的話，只要把原來 LED_EFFECT_03 的幀的順序倒轉便可。如果想增加一點點難度，也可以同時把左右方向反轉。



[資源檔案：project-2-assignment-2.2.sb2]

3. 我們嘗試做一個「開合」的效果吧。

首先，讓我們先畫一個草稿。

幀 1：0000110000

幀 2：0001111000

幀 3：0011001100

幀 4：0110000110

幀 5：1100000011

幀 6：0110000110

幀 7：0011001100

幀 8：0001111000

幀 9：0000110000

幀 10：0000000000

然後就計算相應的十進數。

幀 1：0000110000 = 48

幀 2：0001111000 = 120

幀 3：0011001100 = 204

幀 4：0110000110 = 390

幀 5：1100000011 = 771

幀 6：0110000110 = 390

建議答案