

# 113 學年度第1學期慈濟大學醫工系專題研究(一)報告

題目：從認識Arduino板基礎到用Max7219做出16\*16跑馬燈程式碼

---

學生姓名：彭歆惠

學號：113104105

指導教授/單位：楊惠雯

中華民國 113 年 12 月 30 號

# 1. 中文摘要

LED顯示技術已廣泛應用於廣告牌、交通指示及資訊展示等領域。本專題旨在結合Arduino與MAX7219模組，設計一套可用於顯示中文的16×16 LED矩陣模組，成功實現了具有跑馬燈效果的字符展示，探索中文字符動態顯示的技術可行性與應用潛力。

系統設計以Arduino為核心，硬體部分採用MAX7219晶片驅動多個8×8 LED矩陣模組，透過串聯構成16×16的顯示板。我們將10mm LED焊接至經鐫雕機精準打孔的木板上，確保結構穩固並具有美觀效果。同時，設計了合理的電路連接與電源配置，確保系統能長時間穩定運行。

在軟體開發方面，我們設計了字型圖案與多種動態顯示模式，包括字符滾動、閃爍等效果。針對中文字符的特殊性，我們對字形進行了多次優化，確保字符顯示清晰且具有吸引力。開發過程中，不斷進行程式調試與測試，以實現流暢的動態效果並提升系統穩定性。

原計畫預計製作四塊16×16 LED顯示板，分別展示「慈、大、醫、工」四字。然而，由於時間限制與技術挑戰，我們完成了兩塊顯示板，成功實現了「慈、醫」與「大、工」字元的滾動顯示功能。透過多次改進，我們成功突破了單一Arduino控制多矩陣模組的技術難點，使整體系統運行穩定，視覺效果達到了預期目標。

雖然未完全達成原定計畫，本專題證明了利用Arduino與MAX7219控制多矩陣動態顯示的可行性，並在硬體製作、程式設計與系統整合測試中積累了豐富經驗。我們發現了設計中的不足，例如電路布局優化、中文字型細節的精準化，以及顯示效果的多樣性改進，這些為未來的深入研究提供了方向。

整個過程不僅深化了我們對LED顯示技術的理解，也提升了硬體製作與程式開發的實踐能力。此次專題為日後更多創新應用奠定了堅實基礎，展現了結合Arduino與LED矩陣技術在動態顯示領域的廣泛可能性。

## 2. 英文摘要

LED display technology has been widely used in billboards, traffic signs and information displays. In this topic, a set of 16×16 Chinese dynamic display systems is designed and produced by combining Arduino and LED matrix modules, which successfully realizes the character display with marquee effect, and explores the technical feasibility and application potential of Chinese character dynamic display.

The system design is based on Arduino, and the hardware part uses MAX7219 chips to drive multiple 8×8 LED matrix modules, which are connected in series to form a 16×16 display board. We welded 10mm LEDs to the wood panels that are precisely perforated by the laser engraving machine to ensure a stable structure and an aesthetically pleasing effect. At the same time, a reasonable circuit connection and power supply configuration are designed to ensure that the system can run stably for a long time.

In terms of software development, we designed font patterns and a variety of dynamic display modes, including character scrolling, flashing and other effects. In view of the particularity of Chinese characters, we have optimized the glyphs several times to ensure that the characters are displayed clearly and attractively. During the development process, the program is continuously debugged and tested to achieve smooth dynamic effects and improve system stability.

The original plan was to produce four 16×16 LED display boards to display the words "Compassion, Greatness, Medicine and Engineering". However, due to time constraints and technical challenges, we completed two display boards and successfully implemented the scrolling display function of the characters "Compassion, Medicine" and "Da, Gong". Through several improvements, we have successfully broken through the technical difficulties of a single Arduino control multi-matrix module, so that the overall system runs stably and the visual effect has reached the expected goal.

Although the original plan was not fully achieved, this topic proves the feasibility of using Arduino and MAX7219 to control multi-matrix dynamic displays, and has accumulated rich experience in hardware fabrication, programming, and system integration testing. We found deficiencies in the d

esign, such as the optimization of circuit layout, the accuracy of Chinese font details, and the improvement of display diversity, which provided direction for future in-depth research.

The whole process not only deepened our understanding of LED display technology, but also improved the practical ability of hardware production and program development. This session lays a solid foundation for more innovative applications in the future, showing the wide range of possibilities for combining Arduino and LED matrix technology in the field of dynamic displays.

## 3. 前言

### 3.1 背景與動機

Arduino LED顯示大多以小尺寸模組8x8為主，且以顯示英文字母為主要內容，缺乏對中文字顯示的支援和相關教程。

本專題的動機是設計並實現一套從16x16的LED顯示系統，研究中文字動態顯示的技術方法。基於Arduino與MAX7219的數位控制優勢，我們研究如何從現有的8x8模組擴展為8x16模組擴，並進一步整合兩塊8x16模組組成大尺寸顯示系統，實現中文字的動態滾動顯示。

硬體方面，專案選用了8x8的LED矩陣模組作為基本顯示單元，通過焊接與串聯，擴展為8x16，進一步組合成16x16的顯示面板。使用MAX7219作為驅動晶片，大幅簡化硬體接線並實現多模組的協同運作。以Arduino作為核心控制板，負責整體系統的運行，並設計穩定的電路以滿足多模組的供電需求。顯示面板採用經鐫雕機加工的木板製作，精確打孔後安裝LED燈，打造出堅固且美觀的支撐結構。

軟體部分則從基礎功能開發入手，掌握Arduino程式設計，結合LedControl函式庫，簡化了對MAX7219的控制操作。針對16x16顯示屏設計中文字型，將字元進行點陣化編碼。動態效果的開發則聚焦於中文字的滾動顯示，並提供方向、速度與動畫效果的靈活調整，提升系統的表現力與互動性。

## 4. 材料與方法

### 4.1 Arduino開發板開發環境（Arduino IDE）的安裝與配置

Arduino IDE是用於撰寫、編譯和上傳程式到Arduino板上的開發環境

配置步驟如下：

#### 1. 下載與安裝：

從Arduino官網下載適用於作業系統的版本，完成安裝。

#### 2. 連接Arduino：

使用USB線將Arduino UNO連接至電腦，並確認電腦成功識別開發板（驅動自動安裝或手動更新）。

#### 3. 配置開發環境：

在Arduino IDE中選擇「工具 → 開發板 → Arduino UNO」，並在「工具 → 端口」中選擇正確的COM端口。

## 4.2 Max7219模組介紹

### 4.2.1 為什麼要用Max7219驅動晶片

#### 現有8x8 LED的局限性

- 1. 控制困難：單純的8x8 LED矩陣需要直接控制每一列和每一行的通電，導致大量的引腳需求（至少16個），對Arduino的引腳數量有挑戰。
- 2. 編程複雜：手動控制8x8矩陣的每個LED需要設計掃描模式和數據刷新機制，增加程式的複雜性，特別是在多個模組需要同時串在一起時。

#### 使用MAX7219的優勢

- 1. 簡化硬體連接：MAX7219內建掃描與顯示驅動功能，只需三根數據線（**DIN**、**CS**、**CLK**）即可控制8x8 LED矩陣，極大減少Arduino引腳占用。
- 2. 支援模組串聯：多個MAX7219模組可以輕鬆串聯，實現大尺寸矩陣擴展。
- 3. 庫支援廣泛：Arduino社群提供了多個成熟函式庫（如LedControl），簡化了程式開發過程，使文字滾動、圖案設計等功能更容易實現。

### 4.2.2 與Arduino的硬體接線圖

以下為Max7219模組與Arduino的基本接線方式（以一個8x8模組為例）

Max7219	Arduino
VCC	5V
GND	GND
DIN	D12（可自訂）
CS	D11（可自訂）
CLK	D10（可自訂）

## 4.3 LED 8x8跑馬燈實現

### 4.3.1 Arduino程式開發學習

在進入到程式開發前，我們參考 Arduino 最基本的數位引腳控制以及網路的教程 [1]，用來確認 Arduino 與電腦正確連接，以及學會最基本的 LED 控制原理。

#### 4.3.1.1 Led一閃一閃

##### 功能描述：

利用 Arduino 開發板控制連接到數位引腳 8 的 LED，實現 LED 燈每秒閃爍一次的效果。主要通過 `pinMode()` 函式設定引腳模式，並使用 `digitalWrite()` 函式改變引腳的電位高低來控制 LED 的點亮與熄滅。

##### 程式碼：

```
void setup() {  
  
    pinMode(8, OUTPUT); // 設置數位引腳8為輸出  
  
}  
  
void loop() {  
  
    digitalWrite(8, HIGH); // 點亮LED  
  
    delay(1000);           // 延遲1秒  
  
    digitalWrite(8, LOW); // 關閉LED  
  
    delay(1000);           // 延遲1秒  
  
}
```



### 程式碼執行流程圖：

開始



初始化 (setup)

- 設置引腳 8 為輸出模式 (pinMode(8, OUTPUT))



進入主循環 (loop)



高電平輸出 (digitalWrite(8, HIGH))

- 引腳 8 輸出 HIGH (LED 打開)
- 延遲 1 秒 (delay(1000))



低電平輸出 (digitalWrite(8, LOW))

- 引腳 8 輸出 LOW (LED 關閉)
- 延遲 1 秒 (delay(1000))



重複主循環 (loop)

#### 4.3.1.2 sos訊號

##### 功能描述：

利用 Arduino 實現 SOS 摩斯密碼閃爍信號，通過控制 LED 來模擬摩斯密碼的求救信號燈。具體的閃爍模式包括快速閃爍 3 次（每次間隔 0.5 秒）和慢速閃爍 3 次（每次間隔 1 秒）。

##### 程式碼：

```
void setup() {  
  
    pinMode(8, OUTPUT); //設置腳位 8 為輸出，以便控制 LED 的亮滅  
  
}  
  
//loop() 是 Arduino 程式的主循環，會不斷執行裡面的程式碼  
  
void loop() {  
  
    //透過 for 迴圈讓 LED 快速閃爍 3 次  
  
    for(int i=0; i<3; i++){  
  
        digitalWrite(8, HIGH); //開燈  
  
        delay(500); //延遲 500 毫秒  
  
        digitalWrite(8, LOW); // 關燈  
  
        delay(500); //延遲500毫秒  
  
    }  
  
    //讓 LED 慢速閃爍 3 次
```

```
for(int i=0; i<3; i++){  
  
    digitalWrite(8, HIGH); // 開燈  
  
    delay(1000); // 延遲1000毫秒  
  
    digitalWrite(8, LOW); // 關燈  
  
    delay(1000); // 延遲1000毫秒  
  
}  
  
}
```

程式執行流程圖：

開始



初始化 (setup)

設置腳位 8 為輸出模式



進入主循環 (loop)



快速閃爍 LED 3 次



進入快速閃爍迴圈 (i = 0 到 2)

- 開燈 (digitalWrite(8, HIGH))
- 延遲 500 毫秒
- 關燈 (digitalWrite(8, LOW))
- 延遲 500 毫秒

結束快速閃爍迴圈



慢速閃爍 LED 3 次



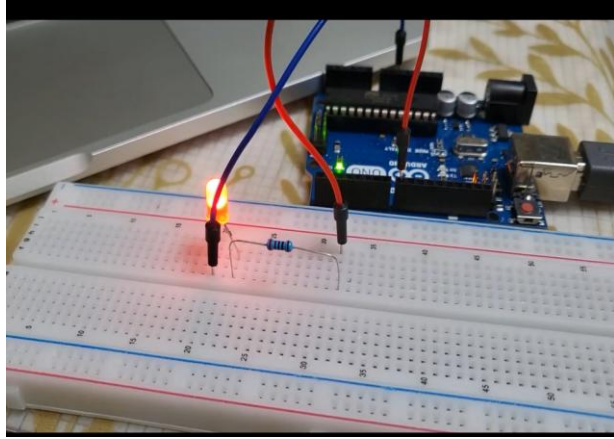
進入慢速閃爍迴圈 (i = 0 到 2)

- 開燈 (digitalWrite(8, HIGH))
- 延遲 1000 毫秒
- 關燈 (digitalWrite(8, LOW))
- 延遲 1000 毫秒

結束慢速閃爍迴圈



重複主循環，從快速閃爍開始

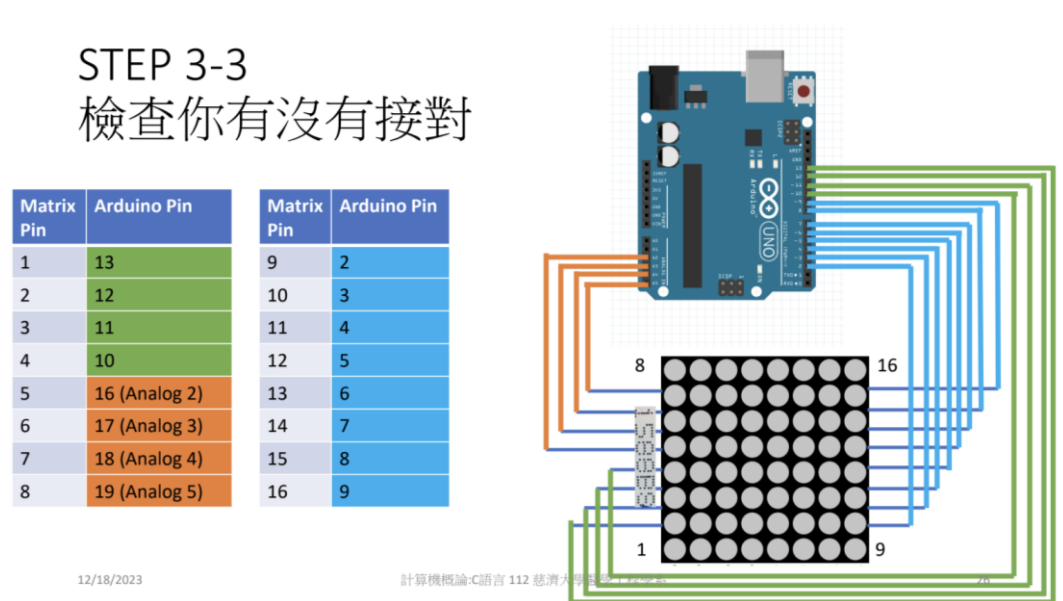


### 4.3.1.3 LED矩陣逐行輪播

#### 功能描述：

利用 Arduino 實現 8x8 LED 矩陣逐行點亮的效果是一個基礎但實用的項目，通過控制矩陣的每一行依次點亮來模擬簡單的動態顯示。具體操作為每行點亮持續 1 秒後熄滅，接著點亮下一行，依序循環重複。

#### 電路圖：



#### 程式碼：

```
//宣告與初始化

//Row（行）：控制 LED 的正極 (陽極)

const int row[8] = {

    2, 7, 13, 5, 19, 12, 18, 10

};

// Column（列）：控制 LED 的負極 (陰極)
```

```
const int col[8] = {  
  
    6, 17, 16, 3, 11, 4, 8, 9  
  
};  
  
void setup() {  
  
    //設置所有行與列腳位為輸出模式  
  
    for (int thisPin = 0; thisPin < 8; thisPin++) {  
  
        pinMode(col[thisPin], OUTPUT); //設定列腳位為輸出模式  
  
        pinMode(row[thisPin], OUTPUT); //設定行腳位為輸出模式  
  
    }  
  
    // 將所有列腳位設為 HIGH，確保 LED 初始狀態全滅（因為沒有對應的 LOW 信號啟動 LED）  
  
    for (int thisPin = 0; thisPin < 8; thisPin++) {  
  
        digitalWrite(col[thisPin], HIGH);  
  
    }  
  
}  
  
// 作用是熄滅所有 LED，將行與列的控制信號全設為 LOW  
  
void clearscreen() {  
  
    for (int y = 0; y < 8; y++) {  
  
        digitalWrite(col[y], LOW);  
  
    }  
  
}
```

```
}

for (int x = 0; x < 8; x++) {

    digitalWrite(row[x], LOW);

}

}

void loop() {

    for(int i = 0; i<8;i++){

        digitalWrite(row[i], HIGH); // 將第 i 行設為 HIGH，點亮該行

        delay(1000); // 延遲 1 秒

        clearscreen(); // 清屏（熄滅所有 LED）

        delay(1000); // 延遲 1 秒

    }

}
```

程式執行流程圖：

開始 (Start)





[初始化 (Setup)]

- 設定所有 Row 與 Column 腳位為輸出模式
- 將所有 Column 設為 HIGH (確保全暗)

↓

進入主循環 (Loop)

↓

[行數迴圈  $i = 0$  至  $7$ ]

↓

- 將第  $i$  行 Row 設為 HIGH (點亮第  $i$  行)
- 延遲 1000 毫秒

↓

- 呼叫清屏函數 `clearscreen()`:

- 所有 Row 設為 LOW

- 所有 Column 設為 LOW

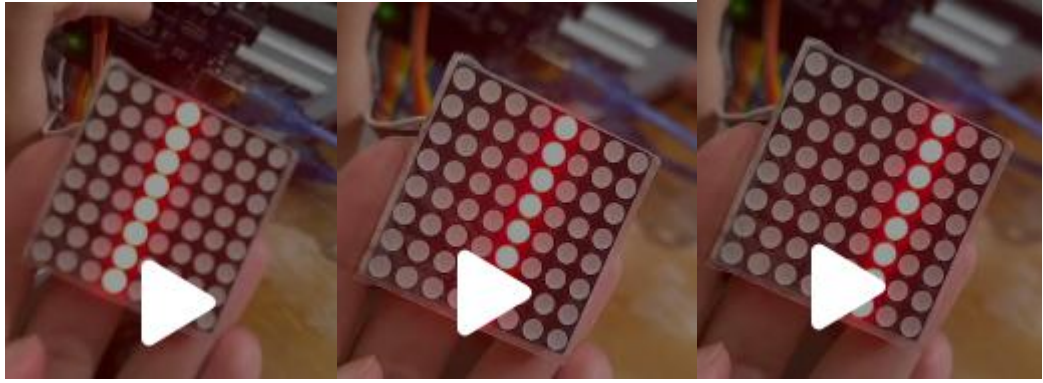
- 延遲 1000 毫秒

↓

[檢查行數迴圈是否完成]

↓ 是 (完成所有行) → 返回主循環

↓ 否 (未完成) → 進入下一行



#### 4.3.1.4 LED矩陣顯示靜態的大字

##### 功能描述：

利用 Arduino 控制 8x8 LED 矩陣，實現逐行亮滅以模擬特定圖案顯示，通過視覺暫留效應使 LED 矩陣呈現靜態圖案效果。該程式控制矩陣的行逐一點亮，同時通過列信號確定哪些 LED 點亮或熄滅，從而組成目標圖案。

##### 程式碼：

```
//宣告與初始化

//Row（行）：控制 LED 的正極 (陽極)

const int row[8] = {

    2, 7, 13, 5, 19, 12, 18, 10

};

// Column（列）：控制 LED 的負極 (陰極)

const int col[8] = {

    6, 17, 16, 3, 11, 4, 8, 9

};

void setup() {

    //設置所有行與列腳位為輸出模式

    for (int thisPin = 0; thisPin < 8; thisPin++) {

        pinMode(col[thisPin], OUTPUT); //設定列腳位為輸出模式
```

```
pinMode(row[thisPin], OUTPUT); //設定行腳位為輸出模式

}

// 將所有列腳位設為 HIGH，確保 LED 初始狀態全滅（因為沒有對應的 LOW 信號
// 啟動 LED）

for (int thisPin = 0; thisPin < 8; thisPin++) {

    digitalWrite(col[thisPin], HIGH);

}

}

// 作用是熄滅所有 LED，將行與列的控制信號全設為 LOW

void clearscreen() {

    for (int y = 0; y < 8; y++) {

        digitalWrite(col[y], LOW);

    }

    for (int x = 0; x < 8; x++) {

        digitalWrite(row[x], LOW);

    }

}
```

```
void loop() {

    digitalWrite(row[0], HIGH); // 啟動第 0 行

    digitalWrite(col[0], HIGH); // 第 0 列不亮

    digitalWrite(col[1], HIGH); // 第 1 列不亮

    digitalWrite(col[2], HIGH); // 第 2 列不亮

    digitalWrite(col[3], LOW); // 第 3 列亮

    digitalWrite(col[4], HIGH); // 第 4 列不亮

    digitalWrite(col[5], HIGH); // 第 5 列不亮

    digitalWrite(col[6], HIGH); // 第 6 列不亮

    digitalWrite(col[7], HIGH); // 第 7 列不亮

    delay(10); //延遲10毫秒

//函數清空顯示

    clearscren();

    delay(1); //延遲1毫秒

    digitalWrite(row[1], HIGH); // 啟動第 1 行

    digitalWrite(col[0], HIGH); // 第 0 列不亮

    digitalWrite(col[1], HIGH); // 第 1 列不亮

    digitalWrite(col[2], HIGH); // 第 2 列不亮
```

```
digitalWrite(col[3], LOW); // 第 3 列亮

digitalWrite(col[4], HIGH); // 第 4 列不亮

digitalWrite(col[5], HIGH); // 第 5 列不亮

digitalWrite(col[6], HIGH); // 第 6 列不亮

digitalWrite(col[7], HIGH); // 第 7 列不亮

delay(10); //延遲10毫秒

//函數清空顯示

clearscreen();

delay(1); //延遲1毫秒


digitalWrite(row[2], HIGH);// 啟動第 2 行

digitalWrite(col[0], LOW); // 第 0 列亮

digitalWrite(col[1], LOW); // 第 1 列亮

digitalWrite(col[2], LOW); // 第 2 列亮

digitalWrite(col[3], LOW); // 第 3 列亮

digitalWrite(col[4], LOW); // 第 4 列亮

digitalWrite(col[5], LOW); // 第 5 列亮

digitalWrite(col[6], LOW); // 第 6 列亮

digitalWrite(col[7], LOW); // 第 7 列亮
```

```
delay(10); //延遲10毫秒
```

```
//函數清空顯示
```

```
clearscreen();
```

```
delay(1); //延遲1毫秒
```

```
digitalWrite(row[3], HIGH); // 啟動第 3 行
```

```
digitalWrite(col[0], HIGH); // 第 0 列不亮
```

```
digitalWrite(col[1], HIGH); // 第 1 列不亮
```

```
digitalWrite(col[2], HIGH); // 第 2 列不亮
```

```
digitalWrite(col[3], LOW); // 第 3 列亮
```

```
digitalWrite(col[4], HIGH); // 第 4 列不亮
```

```
digitalWrite(col[5], HIGH); // 第 5 列不亮
```

```
digitalWrite(col[6], HIGH); // 第 6 列不亮
```

```
digitalWrite(col[7], HIGH); // 第 7 列不亮
```

```
delay(10); //延遲10毫秒
```

```
//函數清空顯示
```

```
clearscreen();
```

```
delay(1); //延遲1毫秒
```

```
digitalWrite(row[4], HIGH); // 啟動第 4 行
```

```
digitalWrite(col[0], HIGH); // 第 0 列不亮
```

```
digitalWrite(col[1], HIGH); // 第 1 列不亮
```

```
digitalWrite(col[2], LOW); // 第 2 列亮
```

```
digitalWrite(col[3], HIGH); // 第 3 列不亮
```

```
digitalWrite(col[4], LOW); // 第 4 列亮
```

```
digitalWrite(col[5], HIGH); // 第 5 列不亮
```

```
digitalWrite(col[6], HIGH); // 第 6 列不亮
```

```
digitalWrite(col[7], HIGH); // 第 7 列不亮
```

```
delay(10); //延遲10毫秒
```

```
//函數清空顯示
```

```
clearscreen();
```

```
delay(1); //延遲1毫秒
```

```
digitalWrite(row[5], HIGH); // 啟動第 5 行
```

```
digitalWrite(col[0], HIGH); // 第 0 列不亮
```

```
digitalWrite(col[1], LOW); // 第 1 列亮
```

```
digitalWrite(col[2], HIGH); // 第 2 列不亮
```

```
digitalWrite(col[3], HIGH); // 第 3 列不亮
```



```
digitalWrite(col[4], HIGH); // 第 4 列不亮
```

```
digitalWrite(col[5], LOW); // 第 5 列亮
```

```
digitalWrite(col[6], HIGH); // 第 6 列不亮
```

```
digitalWrite(col[7], HIGH); // 第 7 列不亮
```

```
delay(10); //延遲10毫秒
```

```
//函數清空顯示
```

```
clearscreen();
```

```
delay(1); //延遲1毫秒
```

```
digitalWrite(row[6], HIGH); // 啟動第 6 行
```

```
digitalWrite(col[0], LOW); // 第 0 列亮
```

```
digitalWrite(col[1], HIGH); // 第 1 列不亮
```

```
digitalWrite(col[2], HIGH); // 第 2 列不亮
```

```
digitalWrite(col[3], HIGH); // 第 3 列不亮
```

```
digitalWrite(col[4], HIGH); // 第 4 列不亮
```

```
digitalWrite(col[5], HIGH); // 第 5 列不亮
```

```
digitalWrite(col[6], LOW); // 第 6 列亮
```

```
digitalWrite(col[7], HIGH); // 第 7 列不亮
```

```
delay(10); //延遲10毫秒
```

```
//函數清空顯示
```

```
clearscreen();
```

```
delay(1); //延遲1毫秒
```

```
}
```

### 程式執行流程圖：

開始



初始化 (setup)：

設置行腳位和列腳位為輸出模式

將所有列腳位設為 HIGH，確保 LED 初始狀態為熄滅



進入主循環 (loop)



執行每行的顯示循環：

顯示第 0 行：

- 開啟第 0 行 (row[0])

- 設定第 3 列為 LOW (點亮)，其他列為 HIGH (熄滅)

- 延遲 10 毫秒

- 清空顯示（清除所有行和列的信號）

- 延遲 1 毫秒

↓

顯示第 1 行：

- 開啟第 1 行 (row[1])

- 設定第 3 列為 LOW，其他列為 HIGH

- 延遲 10 毫秒

- 清空顯示（清除所有行和列的信號）

- 延遲 1 毫秒

↓

顯示第 2 行：

- 開啟第 2 行 (row[2])

- 設定第 0 到第 7 列為 LOW（全部點亮）

- 延遲 10 毫秒

- 清空顯示（清除所有行和列的信號）

- 延遲 1 毫秒

↓

顯示第 3 行：

- 開啟第 3 行 (row[3])

- 設定第 3 列為 LOW，其他列為 HIGH
- 延遲 10 毫秒
- 清空顯示（清除所有行和列的信號）
- 延遲 1 毫秒



顯示第 4 行：

- 開啟第 4 行 (row[4])
- 設定第 2 和第 4 列為 LOW，其他列為 HIGH
- 延遲 10 毫秒
- 清空顯示（清除所有行和列的信號）
- 延遲 1 毫秒



顯示第 5 行：

- 開啟第 5 行 (row[5])
- 設定第 1 和第 5 列為 LOW，其他列為 HIGH
- 延遲 10 毫秒
- 清空顯示（清除所有行和列的信號）
- 延遲 1 毫秒



顯示第 6 行：

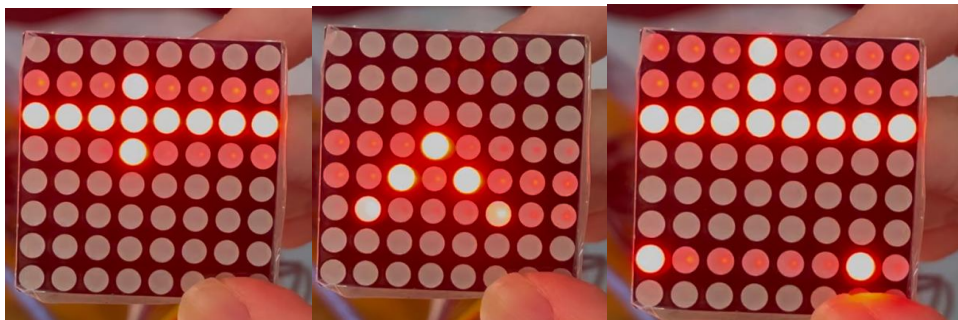
- 開啟第 6 行 (row[6])
- 設定第 0 和第 6 列為 LOW，其他列為 HIGH
- 延遲 10 毫秒
- 清空顯示（清除所有行和列的信號）
- 延遲 1 毫秒

↓

返回循環，繼續顯示下一行

↓

循環結束後，重新開始顯示（從第 0 行開始）



#### 4.3.1.5 LED矩陣顯示靜態的大字—改用大字01位元表示法

##### 功能描述：

利用 Arduino 實現 8x8 LED 矩陣顯示字元「大」，可以通過位元運算定義圖案的顯示結構。使用行的逐步掃描刷新，配合列的開關狀態控制，最終使 LED 矩陣呈現靜態的「大」字效果。

##### 程式碼：

```
//宣告與初始化

//Row（行）：控制 LED 的正極 (陽極)

const int row[8] = {

    2, 7, 13, 5, 19, 12, 18, 10

};

// Column（列）：控制 LED 的負極 (陰極)





const int col[8] = {

    6, 17, 16, 3, 11, 4, 8, 9

};

// '大'字的8x8圖案表示

byte character[8] = {

    0b00010000, // 
    0b00010000, // 
    0b00010000, // 
    0b11111111, // 
```

```

0b00010000, // □ □ □ □ ■ □ □ □
0b00101000, // □ □ □ ■ □ ■ □ □
0b01000100, // □ □ ■ □ □ □ ■ □
0b10000010 // □ ■ □ □ □ □ □ ■
};

void setup() {

  //設置所有行與列腳位為輸出模式

  for (int thisPin = 0; thisPin < 8; thisPin++) {

    pinMode(col[thisPin], OUTPUT); //設定列腳位為輸出模式

    pinMode(row[thisPin], OUTPUT); //設定行腳位為輸出模式

  }

  // 將所有列腳位設為 HIGH，確保 LED 初始狀態全滅（因為沒有對應的 LOW 信號
  啟動 LED）

  for (int thisPin = 0; thisPin < 8; thisPin++) {

    digitalWrite(col[thisPin], HIGH);

  }

}

void loop() {

  // 持續刷新顯示：這段程式碼會不斷循環，刷新顯示「大」字圖案

```

```
for (int i = 0; i < 8; i++) {  
  
    // 開啟當前行：在每次循環中，會依次顯示每一行  
  
    digitalWrite(row[i], HIGH);  
  
    // 寫入當前行的圖案：根據 character[i] 的值，控制該行的每顆 LED 的開關  
  
    for (int j = 0; j < 8; j++) {  
  
        if (character[i] & (1 << j)) {  
  
            // 如果該位元為 1，則關閉該列，變為低電位  
  
            digitalWrite(col[j], LOW);  
  
        } else {  
  
            // 如果該位元為 0，則保持該列為高電位 (LED 關閉)  
  
            digitalWrite(col[j], HIGH);  
  
        }  
  
    }  
  
    // 每行之間的小延遲以控制刷新速度  
  
    delay(1); // 減少延遲以提高刷新速度  
  
    // 顯示完畢後關閉該行，防止多行同時顯示  
  
    digitalWrite(row[i], LOW);  
  
}
```



```
}
```

程式執行流程圖：

開始



進入主循環 (loop)



設置行循環 (i = 0 至 7)



|-- 是 (i < 8)?



|     開啟當前行 (digitalWrite(row[i], HIGH))



|     設置列循環 (j = 0 至 7)



|     |-- 是 (j < 8)?



|     |     檢查位元 (character[i] & (1 << j))



| | | -- 是 (該位元為 1)?

| | | ↓

| | | 關閉列 (digitalWrite(col[j], LOW))

| | | ↓

| | | 繼續下列循環

| | ↓

| | | -- 否 (該位元為 0)

| | ↓

| | 保持列為高 (digitalWrite(col[j], HIGH))

| | ↓

| | 繼續下列循環

| ↓

| 設置列循環結束 (j++)

| ↓

| 等待 (delay(1)) // 控制刷新速度

| ↓

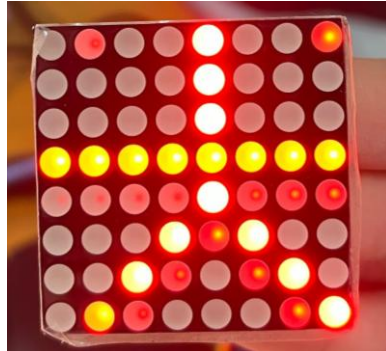
| 關閉當前行 (digitalWrite(row[i], LOW))

| ↓

| 行循環結束 (i++)



否 ( $i \geq 8$ ) → 返回主循環



## 4.4 Max7219 16×16跑馬燈實現

### 4.4.1 程式庫安裝

MAX7219 晶片需透過相關的程式庫驅動在 Arduino 整合環境中安裝程式庫的方式，請參考網路教材 [2]

### 4.4.2 Max7219 程式學習開發

#### 4.4.2.1 Max7219一點亮特定的行列

這段程式碼透過 LedControl 函式庫，控制 MAX7219 晶片來操作 LED 矩陣，實現點亮指定位置的 LED 燈（例如第 0 行第 0 列和第 1 行第 1 列），並以簡單的閃爍效果展示功能。

程式碼：

```
#include <LedControl.h>

int DIN = 12;

int CS = 11;

int CLK = 10;

// 初始化 LedControl 物件，指定 DIN, CLK, CS 和 MAX7219 的數量（這裡是 1）
LedControl lc = LedControl(DIN, CLK, CS, 1);

void setup() {

    // 啟動顯示器

    lc.shutdown(0, false); // 關閉省電模式，啟動顯示器

    lc.setIntensity(0, 8); // 設定亮度，範圍 0-15
```

```
lc.clearDisplay(0);    // 清除顯示  
  
}  
  
void loop() {  
  
    // 顯示圖案測試 (第1行第1列的LED開啟)  
  
    lc.setLed(0, 0, 0, true); // 第0個顯示器, 第0行, 第0列, 打開  
  
    delay(500); //延遲500毫秒  
  
  
    // 顯示另一個LED (第1行第1列的LED開啟)  
  
    lc.setLed(0, 1, 1, true); // 第0個顯示器, 第1行, 第1列, 打開  
  
    delay(500); //延遲500毫秒  
  
  
    // 清除顯示  
  
    lc.clearDisplay(0);  
  
    delay(500); //延遲500毫秒  
  
}
```

**程式碼執行流程圖：**

開始



初始化 (setup)

- 啟動 LED 顯示器 (lc.shutdown)
- 設定亮度為 8 (lc.setIntensity)
- 清空顯示畫面 (lc.clearDisplay)



進入主循環 (loop)



開啟第 0 行第 0 列的 LED

- lc.setLed(0, 0, 0, true)
- 延遲 500 毫秒



開啟第 1 行第 1 列的 LED

- lc.setLed(0, 1, 1, true)
- 延遲 500 毫秒

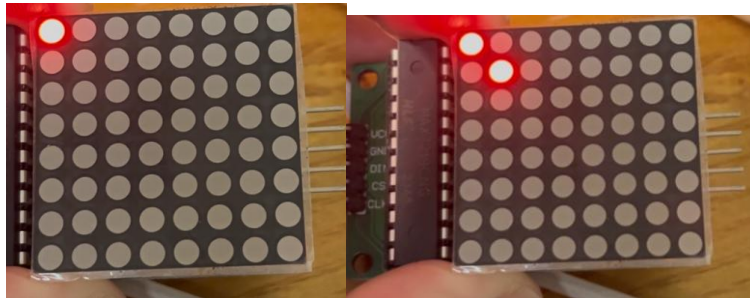


清空顯示畫面

- lc.clearDisplay(0)
- 延遲 500 毫秒



重複主循環，從第 0 行第 0 列的 LED 開始



#### 4.4.2.2 Max7219逐排點亮大字

功能描述：

這段程式碼使用 **LedControl 函式庫**，控制 MAX7219 晶片，點亮指定位置的 LED，依序顯示出一組動態圖案。具體效果是 LED 矩陣依次點亮不同的燈，呈現一個動態的「建築逐層升起的效果」。

程式碼：

```
#include <LedControl.h>

int DIN = 12;

int CS = 11;

int CLK = 10;

int d = 500;

// 初始化 LedControl 物件，指定 DIN, CLK, CS 和 MAX7219 的數量（這裡是 1）
LedControl lc = LedControl(DIN, CLK, CS, 1);

void setup() {
```

```
// 啟動顯示器

lc.shutdown(0, false); // 關閉省電模式，啟動顯示器

lc.setIntensity(0, 8); // 設定亮度，範圍 0-15

lc.clearDisplay(0); // 清除顯示
}

void loop() {

lc.setLed(0, 0, 3, true);

delay(d);

lc.setLed(0, 1, 3, true);

delay(d);

lc.setLed(0, 2, 0, true);

lc.setLed(0, 2, 1, true);

lc.setLed(0, 2, 2, true);

lc.setLed(0, 2, 3, true);

lc.setLed(0, 2, 4, true);

lc.setLed(0, 2, 5, true);

lc.setLed(0, 2, 6, true);
```



```
lc.setLed(0, 2, 7, true);
```

```
delay(d);
```

```
lc.setLed(0, 3, 3, true);
```

```
delay(d);
```

```
lc.setLed(0, 4, 2, true);
```

```
lc.setLed(0, 4, 4, true);
```

```
delay(d);
```

```
lc.setLed(0, 5, 1, true);
```

```
lc.setLed(0, 5, 5, true);
```

```
delay(d);
```

```
lc.setLed(0, 6, 0, true);
```

```
lc.setLed(0, 6, 6, true);
```

```
delay(d);
```

```
// 清除顯示
```

```
lc.clearDisplay(0);
```

```
delay(d);  
  
}
```

程式碼執行流程圖：

開始



初始化 (setup)

- 啟動 LED 顯示器 (lc.shutdown)
- 設定亮度為 8 (lc.setIntensity)
- 清空顯示畫面 (lc.clearDisplay)



進入主循環 (loop)



顯示 LED 點亮動作

點亮第 0 行第 3 列 (lc.setLed(0, 0, 3, true))

- 延遲 500 毫秒



點亮第 1 行第 3 列 (lc.setLed(0, 1, 3, true))

- 延遲 500 毫秒



點亮第 2 行的所有 LED

- 點亮 (0, 2, 0) 至 (0, 2, 7)
- 延遲 500 毫秒



點亮第 3 行第 3 列 (lc.setLed(0, 3, 3, true))

- 延遲 500 毫秒



點亮第 4 行的第 2 和第 4 列

- (lc.setLed(0, 4, 2, true), lc.setLed(0, 4, 4, true))
- 延遲 500 毫秒



點亮第 5 行的第 1 和第 5 列

- (lc.setLed(0, 5, 1, true), lc.setLed(0, 5, 5, true))
- 延遲 500 毫秒



點亮第 6 行的第 0 和第 6 列

- (lc.setLed(0, 6, 0, true), lc.setLed(0, 6, 6, true))
- 延遲 500 毫秒



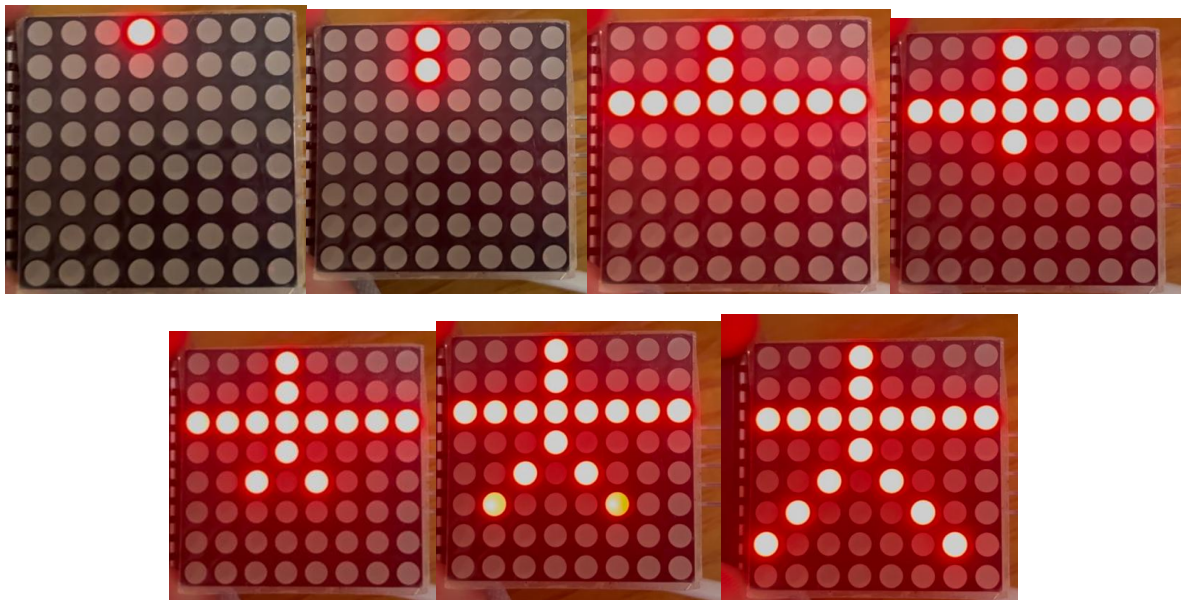
清空顯示畫面

- lc.clearDisplay(0)

- 延遲 500 毫秒



重複主循環



#### 4.4.2.3 Max7219左右輪播大

##### 功能描述：

在 Arduino 程式中，shift 通常指的是位元操作中的「**移位操作**」，即對數字的二進制位元進行左移 (<<) 或右移 (>>) 的操作。

##### 程式碼：

```
#include <LedControl.h>    LedControl.h 是一個用來控制 MAX7219 驅動 IC 的庫。這個庫簡化了與 8x8 LED 矩陣的控制

//DIN、CS、CLK 分別對應 Arduino 的數位腳位 12、11 和 10，用來與 MAX7219 I C 通訊

int DIN = 12;

int CS = 11;

int CLK = 10;


// 初始化 LedControl 物件，指定 DIN, CLK, CS 和 MAX7219 的數量（這裡是 1）

LedControl lc = LedControl(DIN, CLK, CS, 1);
```

// 8x8 矩陣的「大」字圖形，使用二進制表示（從右至左數，0 代表關閉，1 代表開啟）

```
byte daCharacter[8] = {
```

```
    B00010000,
```

```
    B00010000,
```

```
    B01111110,
```

```
    B00010000,
```

```
    B00101000,
```

```
    B01000100,
```

```
    B10000010,
```

```
    B00000000
```

```
};
```

```
void setup() {
```

```
    lc.shutdown(0, false); // 啟動 LED 模組（如果設為 true，則關閉顯示）
```

```
    lc.setIntensity(0, 8); // 設定亮度 (0-15)
```

```
    lc.clearDisplay(0); // 清空顯示，確保開始時矩陣是空白
```

```
}
```

```
void loop() {  
  
    // 輪播效果 - 從右到左滾動「大」  
  
    //shift 從 0 到 7，控制整個字元圖案從右向左移動。每次移動一位，daCharacter  
    [row] >> shift 會將第 row 行的二進制圖案右移 shift 次，這樣 LED 顯示會有滾動  
    效果  
  
    for (int shift = 0; shift < 8; shift++) {  
  
        //這個迴圈遍歷 8 行的每一行，並將對應的圖案（右移後的位元組）通過 lc.setRow()  
        顯示到 LED 矩陣中。lc.setRow(0, row, data) 表示將第 row 行顯示為 data  
  
        for (int row = 0; row < 8; row++) {  
  
            byte data = daCharacter[row] >> shift; // 將字元右移  
  
            lc.setRow(0, row, data);           // 設定每一行資料  
  
        }  
  
        delay(250); // 控制滾動速度，移動後延遲 250 毫秒  
  
    }  
  
    // 清除顯示後，暫停一下，準備下一次輪播  
  
    lc.clearDisplay(0);  
  
    delay(1000); //延遲1秒  
  
}
```

shift功能講解：

```
for (int row = 0; row < 8; row++) {  
  
    byte data = daCharacter[row] >> shift; // 將字元右移  
  
    lc.setRow(0, row, data);                // 設定每一行資料  
  
}
```

這段程式碼的作用是：

1. 將一個字元的每一行二進制資料（例如 8x8 點陣 LED 矩陣的行資料）右移一定的位數（shift）。
2. 然後用 `lc.setRow()` 函式更新 MAX7219 的某一行 LED。

具體來說：

- `daCharacter[row]` 是一個陣列，存儲了某個字元（或圖案）的 8x8 矩陣資料，每個元素代表矩陣的一行（用 8 位元的二進制數表示）。
- `>> shift` 將該行資料向右移動 `shift` 位，使整個字元向右偏移。
- `lc.setRow(0, row, data)` 將偏移後的資料顯示到 LED 矩陣的某一行。

移位操作的原理：

右移操作（>>）

右移操作會將數字的位元向右移動，類似「整數除以 2 的次方」。

例如：`daCharacter[row] >> 1` 表示將資料的每一位向右移 1 位。

移位結果:右移後，數值會縮小，相當於「位元向右滑動」

移位方向：

>> 向右移，低位溢出，丟掉，並用 0 填補高位。



<< 向左移，低位補 0，並將高位溢出丟掉。

示例

假設有一個字元的某一行資料為 0b11110000：

- 未移位：0b11110000 → 對應顯示效果：####
- 右移 1 位：0b01111000 → 對應顯示效果：###
- 右移 2 位：0b00111100 → 對應顯示效果：##

程式執行流程圖：

開始



初始化 (setup)



啟動 LED 模組 (lc.shutdown(0, false))

設定 LED 亮度 (lc.setIntensity(0, 8))

清空顯示 (lc.clearDisplay(0))



進入主循環 (loop)



輪播顯示「大」字



外部迴圈：控制移動位元 (shift = 0 到 7)



內部迴圈：設定每一行的圖案

- 計算移位後的行圖案 (`data = daCharacter[row] >> shift`)
- 更新 LED 矩陣行資料 (`lc.setRow(0, row, data)`)



暫停 250 毫秒，顯示更新

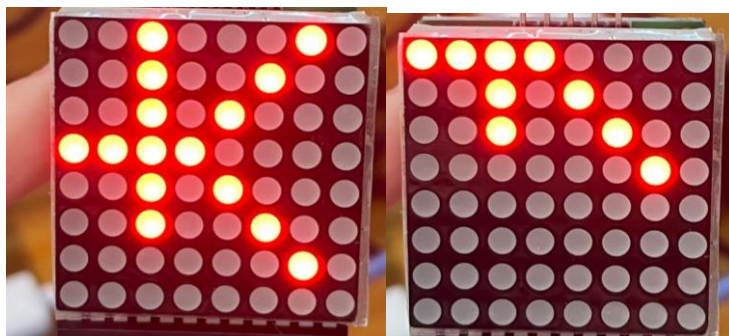


清空顯示 (`lc.clearDisplay(0)`)

暫停 1 秒



返回主循環，重複輪播顯示



#### 4.4.2.4 Max7219上下大輪播

利用 Arduino 和兩個 8x8 LED 矩陣模組，可以模擬一個 8x16 的顯示空間，實現字元「大」的動態滾動效果。這個項目通過使用 LedControl 函式庫管理 MAX7219，實現對 LED 矩陣的串列控制，簡化了硬體驅動的複雜性。在程式中，通過定義字元「大」的 16 行位元資料，利用 for 迴圈實現垂直滾動效果。為了使圖案在兩個 LED 矩陣模組間無縫銜接，加入了邏輯判斷，確保圖案能夠平滑過渡。offset 是控制滾動的關鍵變數，它決定了圖案在 LED 矩陣模組上的顯示位置。隨著 offset 不斷變化，圖案從矩陣的頂部逐行向下滾動，產生動態的視覺效果。

**程式碼：**

```
#include <LedControl.h>
```

```
int DIN = 12;
```

```
int CS = 11;
```

```
int CLK = 10;
```

```
// 初始化 LedControl 物件，指定 DIN, CLK, CS 和 MAX7219 的數量（這裡是 2，表示  
2 個 8x8 模組）
```

```
LedControl lc = LedControl(DIN, CLK, CS, 2);
```

```
// 8x8 矩陣的「大」字圖形，使用二進制表示
```

```
byte daCharacter[8] = {
```

```
    B01000100,
```

```
    B00100100,
```

```
    B00010100,
```

```
    B00001111,
```

```
    B00010100,
```

```
    B00100100,
```

```
    B01000100,
```

```
    B10000100,
```

```
};
```

```
void setup() {
```

```
    // 啟動 LED 模組並設定亮度
```

```
    for (int i = 0; i < 2; i++) {
```

```
        lc.shutdown(i, false); // 啟動 LED 模組
```

```
lc.setIntensity(i, 8); // 設定亮度 (0-15)

lc.clearDisplay(i); // 清空顯示

}

}

void loop() {

    // 向下滾動「大」字的圖案

    for (int offset = -7; offset < 16; offset++) { // offset範圍從 -7 到 15

        // 清空顯示

        lc.clearDisplay(0);

        lc.clearDisplay(1);

        // 在第1個板子顯示上半部（如果 offset 大於等於0，才顯示）

        for (int row = 0; row < 8; row++) {

            int displayRow = row + offset; // 計算當前行

            if (displayRow >= 0 && displayRow < 8) {

                lc.setRow(0, displayRow, daCharacter[row]);

            }

        }

    }

}
```

// 在第2個板子顯示下半部（如果 offset 超出第1個板子的範圍，顯示剩餘部分）

```
for (int row = 0; row < 8; row++) {
```

```
    int displayRow = row + offset - 8; // 調整行數讓圖案在第2個板子上繼續滾動
```

```
    if (displayRow >= 0 && displayRow < 8) {
```

```
        lc.setRow(1, displayRow, daCharacter[row]);
```

```
    }
```

```
}
```

```
delay(250); // 控制滾動速度
```

```
}
```

// 完成滾動後，從下到上逐行隱去圖案

```
for (int row = 7; row >= 0; row--) {
```

```
    // 清除第1個板子上的一行
```

```
    lc.setRow(0, row, 0);
```

```
    // 清除第2個板子上的一行
```

```
    lc.setRow(1, row, 0);
```

```
    delay(250); // 控制隱去速度
```

```
}
```

```
// 清除顯示後，準備下一次顯示

lc.clearDisplay(0);

lc.clearDisplay(1);

delay(1000);

}
```

offset功能講解：

### 1. offset 的作用

offset 是一個整數變數，用來控制圖案在 LED 矩陣模組中的**垂直位置偏移**。隨著 offset 的變化，圖案會在 LED 矩陣模組中逐行上下移動，實現滾動效果。

在程式中，offset 的範圍設置為 -7 到 15，涵蓋了整個滾動過程，包括：

1. 圖案逐漸進入第 1 個模組。
2. 圖案從第 1 個模組滾動到第 2 個模組。
3. 圖案逐漸完全離開第 2 個模組。

### 2. offset 的運算邏輯

在滾動過程中，offset 控制每一行的圖案是否顯示，以及顯示在哪一個模組上。

#### (1) 第 1 個模組的顯示條件

```
int displayRow = row + offset;
if (displayRow >= 0 && displayRow < 8) {
    lc.setRow(0, displayRow, daCharacter[row]);
}
```

- displayRow 是計算圖案行位置的變數，等於圖案的行數 (row) 加上偏移量 (offset)。
- 如果 displayRow 在範圍 0 到 7 之間（對應 LED 矩陣的有效行數），則圖案會顯示在第 1 個模組 (lc.setRow(0, ...))。

例如：

- 當 offset = 0 時，圖案的第 1 行 (row = 0) 顯示在模組的第 1 行 (displayRow = 0)。
- 當 offset = 3 時，圖案的第 1 行顯示在模組的第 4 行 (displayRow = 3)。

## (2) 第 2 個模組的顯示條件

```
int displayRow = row + offset - 8;  
if (displayRow >= 0 && displayRow < 8) {  
    lc.setRow(1, displayRow, daCharacter[row]);  
}
```

- 第 2 個模組的 displayRow 在計算時減去 8，因為第 2 個模組的行數從圖案的第 9 行開始。
- 如果 displayRow 在範圍 0 到 7 之間，圖案會顯示在第 2 個模組 (lc.setRow(1, ...))。

例如：

- 當 offset = 9 時，圖案的第 1 行 (row = 0) 顯示在第 2 個模組的第 1 行 (displayRow = 1)。
- 當 offset = 15 時，圖案的第 7 行 (row = 7) 顯示在第 2 個模組的第 7 行 (displayRow = 7)。



模組 1:

[            ] <- 無顯示

[            ]

[            ]

...

模組 2:

[            ]

[    ●       ] <- row = 0 顯示於第 2 個模組的第 1 行

[            ]

...

### 3. offset 的範圍選擇

● 範圍 -7 到 15 是根據圖案和模組設計的：

- -7 到 -1：圖案的上部從邊緣逐漸進入第 1 個模組。
- 0 到 7：圖案完全顯示在第 1 個模組中。
- 8 到 15：圖案從第 1 個模組滾動到第 2 個模組，並逐漸離開。

## 程式執行流程圖：

開始



初始化 (setup)

- 啟動 LED 模組
- 設定亮度為 8 (lc.setIntensity)
- 清空 2 個 LED 板的顯示 (lc.clearDisplay)



進入主循環 (loop)



**\*\*第一部分：向下滾動「大」字圖案\*\***

- offset 從 -7 遞增到 15：



1. 清空所有顯示

- lc.clearDisplay(0) 與 lc.clearDisplay(1)



2. 計算 offset 與行數 (row + offset)

- 在第 1 塊板子顯示圖案的上半部
- 在第 2 塊板子顯示圖案的下半部

↓

3. 控制滾動速度，延遲 250 毫秒

- 重複直到 `offset = 15`，完成向下滾動

↓

第二部分：從下到上逐行隱去圖案

- `row` 從 7 遞減到 0：

↓

1. 清除第 1 個板子 (`lc.setRow(0, row, 0)`)

2. 清除第 2 個板子 (`lc.setRow(1, row, 0)`)

↓

3. 延遲 250 毫秒

- 重複直到 `row = 0`，完成隱去圖案

↓

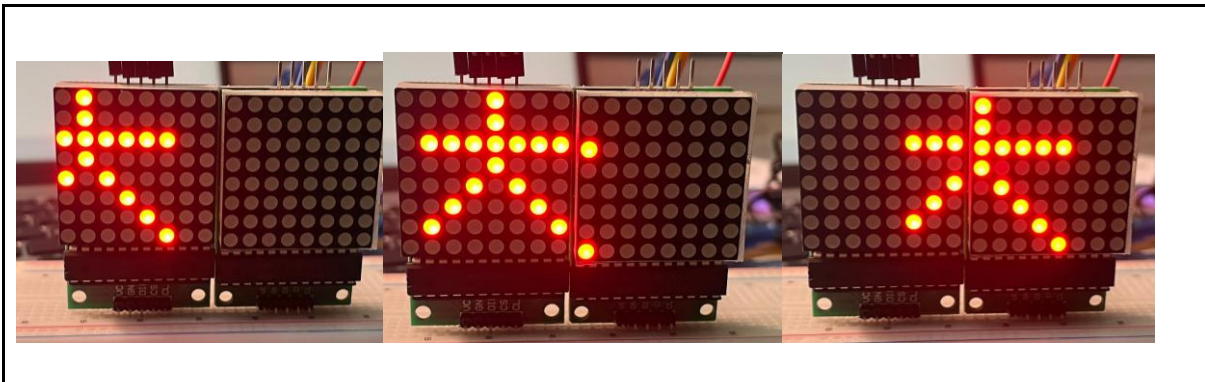
清空顯示並暫停

- `lc.clearDisplay(0)` 與 `lc.clearDisplay(1)`

- 延遲 1000 毫秒，準備下一次顯示

↓

重複主循環



#### 4.4.2.5 慈下半部顯示後向左隱沒

這個項目利用 Arduino 和兩個由 MAX7219 控制的 8x8 LED 點陣模組，串聯構成一個 8x16 的顯示板，實現「慈」字下半部的滾動顯示與消失動態效果。模組透過串列連接，由 Arduino 提供控制訊號，完成字形的動態展示。

程式的主要功能是顯示「慈」字的下半部，並在顯示後實現從左到右的逐步隱沒效果。隱沒過程通過逐列熄滅字形來完成，模擬滾動的動態視覺效果。當滾動完成後，屏幕會被清空，並重新開始滾動，形成循環動作。

程式碼：

```
#include <LedControl.h>
```

```
int DIN = 12;
```

```
int CS = 11;
```

```
int CLK = 10;
```

```
// 初始化 LedControl 物件，指定 DIN, CLK, CS 和 MAX7219 的數量（這裡是 2，表示  
2 個 8x8 模組）
```

```
LedControl lc = LedControl(DIN, CLK, CS, 2);
```

```
// 16 行的字形，使用二進制表示
```

```
byte daCharacter[16] = {
```

```
    B00000000,
```

```
    B01000010,
```

```
    B00110011,
```

```
    B00000010,
```

```
    B01110010,
```

```
    B10000011,
```

```
    B10000110,
```

```
    B10001000,
```

```
    B10110010,
```

```
B10000011,  
  
B11100010,  
  
B00000010,  
  
B00000011,  
  
B00010110,  
  
B01100000,  
  
B00000000, // 額外的空行，保持 16 行  
};  
  
void setup() {  
  
    // 啟動 LED 模組並設定亮度  
  
    for (int i = 0; i < 2; i++) {  
  
        lc.shutdown(i, false); // 啟動 LED 模組  
  
        lc.setIntensity(i, 8); // 設定亮度 (0-15)  
  
        lc.clearDisplay(i);    // 清空顯示  
  
    }  
}  
  
void loop() {
```

```
// 從左到右滾動字形，從 offset 0 到 16
```

```
for (int offset = 0; offset < 16; offset++) {
```

```
    // 清空顯示
```

```
    lc.clearDisplay(0);
```

```
    lc.clearDisplay(1);
```

```
    // 顯示第一個板子上的字形
```

```
    for (int row = 0; row < 8; row++) {
```

```
        if (offset + row < 16) {
```

```
            lc.setRow(0, row, daCharacter[offset + row]); // 在第一個板子顯示字形
```

```
        }
```

```
    }
```

```
    // 第二個板子顯示的範圍（第二個板子顯示字形，並從左滾動到右）
```

```
    for (int row = 0; row < 8; row++) {
```

```
        int displayRow = row + offset; // 這裡顯示字形並移動
```

```
        if (displayRow < 8) {
```

```
            lc.setRow(1, row, daCharacter[displayRow + 8]); // 在第二個板子顯示字形
```

```
        }
```

```
}

// 第一個板子接收第二個板子滾動的內容

for (int row = 0; row < 8; row++) {

    int displayRow = row + offset - 8; // 顯示第二個板子滾動過來的內容

    if (displayRow >= 0 && displayRow < 8) {

        lc.setRow(0, row, daCharacter[displayRow + 8]); // 在第一個板子顯示字形

    }

}

delay(200); // 控制滾動速度

}

// 完成滾動後，清空顯示，準備重新顯示字形

lc.clearDisplay(0);

lc.clearDisplay(1);

delay(1000); // 等待1秒鐘，然後重新顯示

}
```



## 程式執行流程圖：

開始



初始化 (setup)

- 啟動 LED 顯示模組
- 第1個顯示器 (板子)
  - lc.shutdown -> 啟動
  - lc.setIntensity -> 設定亮度
  - lc.clearDisplay -> 清空顯示
- 第2個顯示器 (板子)
  - lc.shutdown -> 啟動
  - lc.setIntensity -> 設定亮度
  - lc.clearDisplay -> 清空顯示



進入主循環 (loop)



從左到右滾動字形 (for offset = 0; offset < 16; offset++)

- 清空第1個和第2個顯示器



顯示第1個板子內容

- 從 daCharacter 中獲取對應的內容行
- 顯示在第1個板子上的行（如果行數有效）

↓

顯示第2個板子內容

- 從 daCharacter 中獲取第2部分內容行
- 顯示在第2個板子上的行（如果行數有效）

↓

將第2個板子的內容傳遞到第1個板子

- 根據 offset 計算並在第1個板子顯示剩餘行

↓

延遲 200 毫秒

（重複上述步驟直到 offset = 16）

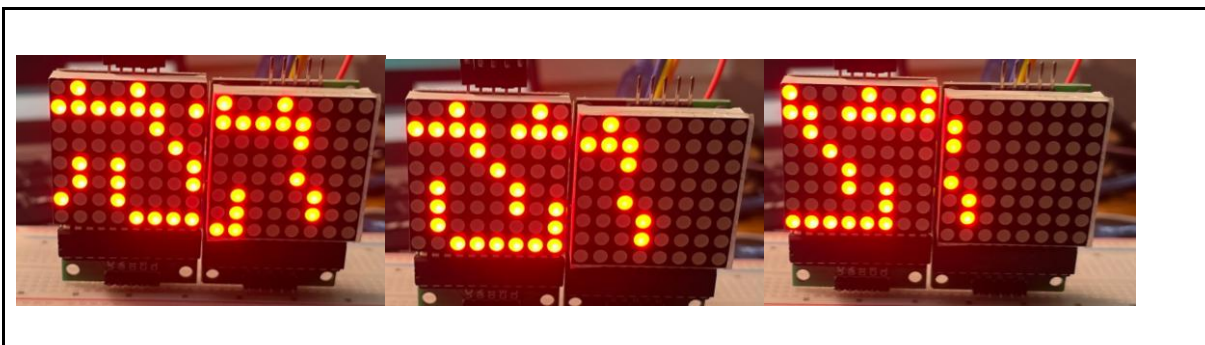
↓

滾動完成

- 清空第1個和第2個顯示器
- 延遲 1 秒

↓

重複主循環



#### 4.4.2.6 慈下半完整

顯示一個 16 行的圖案（daCharacter 字形數據）。

字形從右到左逐行滾動顯示，並在滾動完成後清空模組。

程式碼：

```
#include <LedControl.h>
```

```
int DIN = 12;
```

```
int CS = 11;
```

```
int CLK = 10;
```

```
// 初始化 LedControl 物件，指定 DIN, CLK, CS 和 MAX7219 的數量
```

```
LedControl lc = LedControl(DIN, CLK, CS, 2);
```

```
// 16行的字形
```

```
byte daCharacter[16] = {
```

```
    B00000000,
```

```
    B01000010,
```

```
    B00110011,
```

```
    B00000010,
```

```
    B01110010,
```

```
    B10000011,
```

```
    B10000110,
```

```
    B10001000,
```

```
    B10110010,
```

```
B10000011,  
  
B11100010,  
  
B00000010,  
  
B00000011,  
  
B00010110,  
  
B01100000,  
  
B00000000  
};  
  
void setup() {  
  
    // 啟動 LED 模組並設定亮度  
  
    for (int i = 0; i < 2; i++) { // 總共有 2 個 MAX  
  
        lc.shutdown(i, false); // 啟動 LED 模組，開啟  
  
        lc.setIntensity(i, 8); // 設定亮度  
  
        lc.clearDisplay(i);    // 清空顯示  
  
    }  
}  
  
void loop() {
```

```
// 從右到左滾動字形，從 offset 15 到 -16

for (int offset = 15; offset >= -15; offset--) { // 讓字形一行一行地往左移動

    lc.clearDisplay(0); // 清空第一個

    lc.clearDisplay(1); // 清空第二個


    // 顯示兩個板子的字形，從右到左滾動

    for (int row = 0; row < 8; row++) { // 逐行顯示8行

        // 第一個板子顯示 16x8 字形的左半部分

        if (offset + row < 16 && offset + row >= 0) { // 確保偏移量在有效範圍內

            lc.setRow(0, row, daCharacter[offset + row]); // 在第一個板子上顯示對應行

        }


        // 第二個板子顯示 16x8 字形的右半部分

        if (offset + row + 8 < 16 && offset + row + 8 >= 0) { // 確保偏移量在有效範圍內

            lc.setRow(1, row, daCharacter[offset + row + 8]); // 在第二個板子上顯示對應行

        }

    }

    delay(200); // 控制滾動速度，這裡是 200 毫秒
```

```
}

// 完成滾動後，清空顯示，準備重新顯示字形

lc.clearDisplay(0); // 清空第一個板子的顯示

lc.clearDisplay(1); // 清空第二個板子的顯示


// 讓顯示清空一段時間，然後重新開始滾動

delay(1000); // 等待 1 秒鐘，讓顯示停頓一下，然後重新開始滾動

}
```

#### 程式執行流程圖：

開始



初始化 (setup)

- 啟動 LED 顯示模組
- 第1個顯示器（板子）
  - lc.shutdown(0, false) -> 開啟顯示模組
  - lc.setIntensity(0, 8) -> 設定亮度
  - lc.clearDisplay(0) -> 清空顯示

- 第2個顯示器（板子）
- `lc.shutdown(1, false)` -> 開啟顯示模組
- `lc.setIntensity(1, 8)` -> 設定亮度
- `lc.clearDisplay(1)` -> 清空顯示

↓

進入主循環 (loop)

↓

從右到左滾動字形 (for offset = 15 到 -15)

- 清空第1個和第2個顯示器
- `lc.clearDisplay(0)`
- `lc.clearDisplay(1)`

↓

滾動顯示邏輯

- 循環逐行更新兩個顯示器 (for row = 0 到 7)
- 計算顯示的行數和內容
  - 如果 `offset + row` 在範圍內，顯示在第1個顯示器 (`lc.setRow(0, row)`)
  - 如果 `offset + row + 8` 在範圍內，顯示在第2個顯示器 (`lc.setRow(1, row)`)
- 延遲 200 毫秒以控制滾動速度

↓



完成滾動 (offset = -16)

- 清空第1個和第2個顯示器

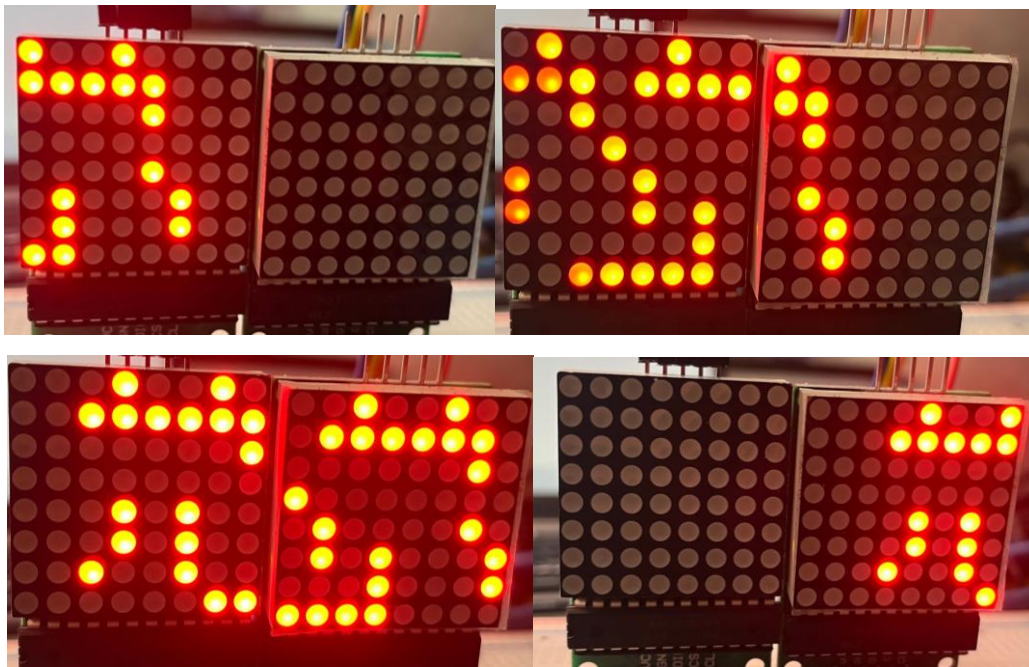
- `lc.clearDisplay(0)`

- `lc.clearDisplay(1)`

- 停頓 1 秒 (`delay(1000)`)

↓

重複主循環 (loop)



#### 4.4.2.7 字分成上下部分

此程式以 Arduino UNO 為核心，結合兩組 MAX7219 晶片控制 4 個 8x8 LED 矩陣模組（兩塊板），實現 16x16 字形的動態滾動顯示功能。

程式碼：

```
#include <LedControl.h>

int DIN = 12;

int CS = 11;

int CLK = 10;


int DIN2 = 7;

int CS2 = 8;

int CLK2 = 9;


// 初始化 LedControl 物件，指定 DIN, CLK, CS 和 MAX7219 的數量

LedControl lc = LedControl(DIN, CLK, CS, 2);

LedControl lc2 = LedControl(DIN2, CLK2, CS2, 2);


// 16行的字形

byte daCharacter1[16] = {

    B00000100,

    B01000100,

    B01100100,
```

```
B11011100,
```

```
B01000101,
```

```
B00100110,
```

```
B00000100,
```

```
B00000100,
```

```
B01000100,
```

```
B01100110,
```

```
B11011101,
```

```
B01000100,
```

```
B00100100,
```

```
B00000100,
```

```
B00000100,
```

```
B00000000
```

```
};
```

```
byte daCharacter2[16] = {
```

```
B00000000,
```

```
B01000010,
```

```
B00110011,
```

```
B00000010,  
  
B01110010,  
  
B10000011,  
  
B10000110,  
  
B10001000,  
  
B10110010,  
  
B10000011,  
  
B11100010,  
  
B00000010,  
  
B00000011,  
  
B00010110,  
  
B01100000,  
  
B00000000  
};  
  
void setup() {  
  
    // 啟動 LED 模組並設定亮度  
  
    for (int i = 0; i < 2; i++) { // 總共有 2 個 MAX  
  
        lc.shutdown(i, false); // 啟動 LED 模組，開啟
```

```

    lc.setIntensity(i, 8); // 設定亮度

    lc.clearDisplay(i);    // 清空顯示
}

for (int i = 0; i < 2; i++) { // 總共有 2 個 MAX

    lc.shutdown(i, false); // 啟動 LED 模組，開啟

    lc.setIntensity(i, 8); // 設定亮度

    lc.clearDisplay(i);    // 清空顯示
}
}

void loop() {

    // 從右到左滾動字形，從 offset 15 到 -16

    for (int offset = 15; offset >= -15; offset--) { // 讓字形一行一行地往左移動

        lc.clearDisplay(0); // 清空第一個

        lc.clearDisplay(1); // 清空第二個


        // 顯示兩個板子的字形，從右到左滾動

        for (int row = 0; row < 8; row++) { // 逐行顯示8行

            // 第一個板子顯示 16x8 字形的左半部分

```

```
if (offset + row < 16 && offset + row >= 0) { // 確保偏移量在有效範圍內

    lc.setRow(0, row, daCharacter1[offset + row]); // 在第一個板子上顯示對應行

    lc2.setRow(0, row, daCharacter2[offset + row]); // 在第一個板子上顯示對應行

}

// 第二個板子顯示 16x8 字形的右半部分

if (offset + row + 8 < 16 && offset + row + 8 >= 0) { // 確保偏移量在有效範圍內

    lc.setRow(1, row, daCharacter1[offset + row + 8]); // 在第二個板子上顯示對應行

    lc2.setRow(1, row, daCharacter2[offset + row + 8]); // 在第二個板子上顯示對應行

}

}

delay(200); // 控制滾動速度，這裡是 200 毫秒

}

// 完成滾動後，清空顯示，準備重新顯示字形

lc.clearDisplay(0); // 清空第一個板子的顯示

lc.clearDisplay(1); // 清空第二個板子的顯示

lc2.clearDisplay(0); // 清空第一個板子的顯示
```

```
lc2.clearDisplay(1); // 清空第二個板子的顯示
```

```
// 讓顯示清空一段時間，然後重新開始滾動
```

```
delay(1000); // 等待 1 秒鐘，讓顯示停頓一下，然後重新開始滾動
```

```
}
```

### 程式執行流程圖：

開始



初始化 (setup)

- 設置第一組 LED 板 (lc)
- 啟動 LED 模組 (shutdown)
- 設置亮度 (setIntensity)
- 清空顯示 (clearDisplay)
- 設置第二組 LED 板 (lc2)
- 啟動 LED 模組 (shutdown)
- 設置亮度 (setIntensity)
- 清空顯示 (clearDisplay)



進入主循環 (loop)



滾動顯示開始 (for offset = 15 到 -15)



清空所有 LED 顯示 (lc 和 lc2)

- lc.clearDisplay(0) 和 lc.clearDisplay(1)
- lc2.clearDisplay(0) 和 lc2.clearDisplay(1)

↓

顯示第一組 LED 板 (lc)

- 從右到左逐行顯示字形左半部分 (daCharacter1)
  - 判斷 offset 是否在有效範圍內，顯示左半部分到第一塊板
  - 判斷 offset + 8 是否在有效範圍內，顯示右半部分到第二塊板

↓

顯示第二組 LED 板 (lc2)

- 從右到左逐行顯示字形 (daCharacter2)
  - 判斷 offset 是否在有效範圍內，顯示左半部分到第一塊板
  - 判斷 offset + 8 是否在有效範圍內，顯示右半部分到第二塊板

↓

滾動完成，延遲 (delay 200ms)

↓

滾動結束後清空顯示

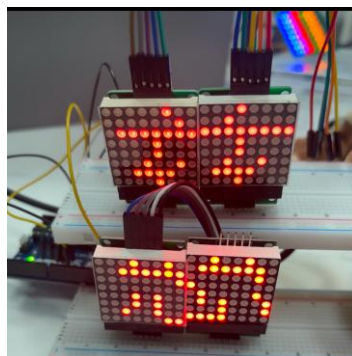
- lc.clearDisplay(0), lc.clearDisplay(1)
- lc2.clearDisplay(0), lc2.clearDisplay(1)

↓

延遲 1 秒 (delay 1000ms)

↓

重複主循環 (loop)





#### 4.4.2.8 完整大、工

在這個項目中，為了支持多模組顯示和滾動效果，系統新增了一組模組（lc2），與原本的 lc 一起控制總共 4 塊 MAX7219 點陣模組，組成一個 16x16 的顯示板。lc 控制第一組模組（前兩塊），lc2 控制第二組模組（後兩塊）。

程式中新增了兩組字形數據（newCharacter1 和 newCharacter2），並將每組字形分為左右兩部分（character1 和 character2）。這樣的結構使得每組字形能夠靈活地分別顯示在模組的兩部分上，實現多字形聯動滾動顯示的功能。

#### 程式碼：

```
#include <LedControl.h>

int DIN = 12;

int CS = 11;

int CLK = 10;


int DIN2 = 7;

int CS2 = 8;

int CLK2 = 9;


// 初始化 LedControl 物件，指定 DIN, CLK, CS 和 MAX7219 的數量
```

```
LedControl lc = LedControl(DIN, CLK, CS, 2);
```

```
LedControl lc2 = LedControl(DIN2, CLK2, CS2, 2);
```

```
// 第一組字形
```

```
byte daCharacter1[16] = {
```

```
    B00000000,
```

```
    B00000110,
```

```
    B00000110,
```

```
    B00000110,
```

```
    B00000110,
```

```
    B00000110,
```

```
    B00000110,
```

```
    B11111110,
```

```
    B11111110,
```

```
    B00000110,
```

```
    B00000110,
```

```
    B00000110,
```

```
    B00000110,
```

```
    B00000110,
```

```
B00000110,
```

```
B00000000
```

```
};
```

```
byte daCharacter2[16] = {
```

```
B00000000,
```

```
B01100000,
```

```
B01100000,
```

```
B01100000,
```

```
B01100000,
```

```
B01100000,
```

```
B01100000,
```

```
B01111111,
```

```
B01111111,
```

```
B01100000,
```

```
B01100000,
```

```
B01100000,
```

```
B01100000,
```

```
B01100000,
```

```
B01100000,
```

```
B00000000
```

```
};
```

```
// 第二組字形
```

```
byte newCharacter1[16] = {
```

```
B00000000,
```

```
B00100000,
```

```
B00100000,
```

```
B00100000,
```

```
B00100000,
```

```
B00100000,
```

```
B00100000,
```

```
B11111111,
```

```
B00100000,
```

```
B00100000,
```

```
B00100000,
```

```
B00100000,
```

```
B00100000,
```

```
B00100000,  
  
B00100000,  
  
B00000000  
};
```

```
byte newCharacter2[16] = {
```

```
    B00000000,  
  
    B10000000,  
  
    B01000000,  
  
    B00100000,  
  
    B00110000,  
  
    B00001100,  
  
    B00000011,  
  
    B00000000,  
  
    B00000011,  
  
    B00001100,  
  
    B00110000,  
  
    B00100000,  
  
    B01000000,
```

```
B10000000,  
  
B00000000,  
  
B00000000,  
  
};  
  
void setup() {  
  
    // 啟動 LED 模組並設定亮度  
  
    for (int i = 0; i < 2; i++) { // 總共有 2 個 MAX  
  
        lc.shutdown(i, false); // 啟動 LED 模組，開啟  
  
        lc.setIntensity(i, 8); // 設定亮度  
  
        lc.clearDisplay(i);    // 清空顯示  
  
    }  
  
    for (int i = 0; i < 2; i++) { // 總共有 2 個 MAX  
  
        lc2.shutdown(i, false); // 啟動 LED 模組，開啟  
  
        lc2.setIntensity(i, 8); // 設定亮度  
  
        lc2.clearDisplay(i);    // 清空顯示  
  
    }  
  
}
```

```
void loop() {  
  
    // 顯示第一組字形 (daCharacter1 和 daCharacter2)  
  
    scrollMessage(daCharacter1, daCharacter2);  
  
  
    // 顯示第二組字形 (newCharacter1 和 newCharacter2)  
  
    scrollMessage(newCharacter1, newCharacter2);  
  
  
    // 完成後清空顯示，準備重新開始滾動  
  
    delay(1000); // 等待 1 秒鐘，讓顯示停頓一下，然後重新開始滾動  
  
}  
  
  
// 滾動顯示字形的函數  
  
void scrollMessage(byte character1[16], byte character2[16]) {  
  
    // 從右到左滾動字形，從 offset 15 到 -16  
  
    for (int offset = 15; offset >= -15; offset--) { // 讓字形一行一行地往左移動  
  
        lc.clearDisplay(0); // 清空第一個  
  
        lc.clearDisplay(1); // 清空第二個  
  
  
        // 顯示兩個板子的字形，從右到左滾動
```

```
for (int row = 0; row < 8; row++) { // 逐行顯示8行
```

```
    // 第一個板子顯示 16x8 字形的左半部分
```

```
    if (offset + row < 16 && offset + row >= 0) { // 確保偏移量在有效範圍內
```

```
        lc.setRow(0, row, character1[offset + row]); // 在第一個板子上顯示對應行
```

```
        lc2.setRow(0, row, character2[offset + row]); // 在第一個板子上顯示對應行
```

```
    }
```

```
    // 第二個板子顯示 16x8 字形的右半部分
```

```
    if (offset + row + 8 < 16 && offset + row + 8 >= 0) { // 確保偏移量在有效範圍內
```

```
        lc.setRow(1, row, character1[offset + row + 8]); // 在第二個板子上顯示對應行
```

```
        lc2.setRow(1, row, character2[offset + row + 8]); // 在第二個板子上顯示對應行
```

```
    }
```

```
}
```

```
delay(200); // 控制滾動速度，這裡是 200 毫秒
```

```
}
```

```
// 完成滾動後，清空顯示
```

```
lc.clearDisplay(0); // 清空第一個板子的顯示
```



```
lc.clearDisplay(1); // 清空第二個板子的顯示  
  
lc2.clearDisplay(0); // 清空第一個板子的顯示  
  
lc2.clearDisplay(1); // 清空第二個板子的顯示  
  
}
```

#### 完整執行流程圖：

開始



初始化 (setup)

- 初始化第1組 LED 板
  - lc.shutdown -> 啟動模組
  - lc.setIntensity -> 設定亮度
  - lc.clearDisplay -> 清空顯示
- 初始化第2組 LED 板
  - lc2.shutdown -> 啟動模組
  - lc2.setIntensity -> 設定亮度
  - lc2.clearDisplay -> 清空顯示



進入主循環 (loop)



顯示第1組字形 (daCharacter1 和 daCharacter2)

- 調用 scrollMessage 函數
- offset 從 15 遞減到 -15
  - 清空第1組顯示器
  - 清空第2組顯示器
  - 逐行顯示對應行
    - 在第1組 LED 顯示左半部分字形
    - 在第2組 LED 顯示右半部分字形
  - 延遲 200 毫秒
- 滾動結束，清空所有顯示



顯示第2組字形 (newCharacter1 和 newCharacter2)

- 調用 scrollMessage 函數
- offset 從 15 遞減到 -15
  - 清空第1組顯示器
  - 清空第2組顯示器
  - 逐行顯示對應行
    - 在第1組 LED 顯示左半部分字形

- 在第2組 LED 顯示右半部分字形

- 延遲 200 毫秒

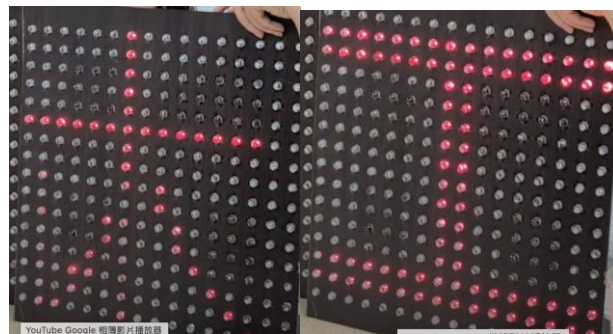
- 滾動結束，清空所有顯示



延遲 1 秒鐘



重複主循環



## 5. 結果

在這次專題中，我們從最基本的LED控制開始，逐步實現了8x8矩陣的圖案顯示，最後完成了16x16中文字滾動的動態展示，整個過程讓我們收穫滿滿。

一開始，我們透過Arduino控制單顆LED，實現了簡單的每秒閃爍效果。接著，設計了摩斯密碼的SOS閃爍模式，快速（0.5秒）和慢速（1秒）交替，讓我們熟悉了程式控制的基本結構和邏輯，對了解Arduino的基礎操作很有幫助，也為後續的矩陣控制奠定了基礎。

接下來，我們挑戰了8x8 LED矩陣，先做了逐行點亮，然後讓矩陣顯示靜態圖案，比如「大」字。為了讓程式更有效率，我們使用了位元運算來設計圖案，這不但讓程式更簡潔，也幫助我們掌握了控制每個LED點亮的邏輯。

隨著專題進一步深入，我們加入了MAX7219晶片，讓矩陣的控制變得更簡單。利用LedControl函式庫，我們實現了指定LED點亮、圖案逐層升起的動畫效果，還模擬了8x16矩陣的滾動顯示。這裡用到了offset這個參數，負責控制圖案的位置，讓滾動效果更順暢。儘管在模組之間的圖案銜接上需要精確計算，結果還是很穩定，方法也很有效。

最後，我們將四個8x8的矩陣組合起來，變成了16x16的顯示系統，成功完成了兩塊16x16的板子，分別展示「慈、大」和「醫、工」的中文字的滾動顯示。為了讓上下部分顯示協調，我們把字型資料分成兩組，分別控制上下兩塊模組。

整體來說，這次專題從簡單的LED控制到16×16矩陣的顯示，難度是一層層提升的，但過程非常有成就感。即使最後因為時間和硬體限制，還有一些地方可以優化，但我們已經證明了這些技術的可行性，也為未來更大規模的應用打下了基礎。

## 6. 討論

透過本次專題，我們成功利用Arduino與MAX7219晶片實現了16×16 LED矩陣中文字的動態顯示，從最基本的LED閃爍到多模組聯動控制，逐步驗證了這些技術的可行性和應用價值。在硬體設計、軟體開發和系統整合的過程中，我們不僅完成了「慈、大」與「醫、工」字元的滾動展示，也累積了對矩陣控制與顯示技術的經驗。

雖然受限於時間，尚未實現一開始所預期製作4塊16×16的LED板子，但這次專題不僅讓我們對相關技術有更深入的理解，也為未來的研究和應用提供了可靠的基礎和方向。

## 7. 參考文獻

1. 《Arduino入門》第一篇：認識Arduino[Internt] Nedlands(TW):傑森創工 Resources: 9 January 2020.[cited 12/22 2024]

Available from:<https://blog.jmaker.com.tw/arduino-tutorials-1/>

2. 【Arduino進階教學課程】MAX7219驅動點陣模組LED Matrix動畫顯示|米羅科技文創學院[Internt] Nedlands(TW):米羅科技有限公司

Resources:2020[cited Der 12/22 2024]

Available from:<https://shop.mirotek.com.tw/tutorial/arduino-max7219/>

