113學年度第1學期慈濟大學醫工系專題研究(一)報告

題目：可編碼、可攜帶之大型LED中文看板之實現：使用MAX7219控制模組進行程式開發

學生姓名：王儷蓁
學號：113104103
指導教授/單位：楊惠雯

中 華 民 國　　年　　月　　日

中文摘要

LED顯示技術在現代生活中應用廣泛，然而，現行開源資料多為顯示英文設計，且使用小型LED矩陣模組。如何製作低成本、具高度可編輯性、可攜帶性的大型LED看板，以供中文使用者自行輸入中文字以變化字型，作為展示、溝通用，仍是未知。因此，本次專題主旨在於探索如何以低成本的方式，自製一個LED矩陣，並且結合Arduino及MAX7219控制模組，顯示中文字跑馬燈。本研究主要負責其中MAX7219控制及中文字跑馬燈效果的實現。首先要先學習基本操作並撰寫程式，使用小型的8x8LED模組針對字型繪製、滾動與閃爍等動態模式進行程式設計，透過反覆測試並改進字形設計與動態顯示效果。最後，套用在大型板上時，再次調整字型的方向、排版及滾動時間，完成整體硬體與軟體整合，確保展示效果穩定。最後成功在兩塊 LED板上分別呈現「慈、醫」與「大、工」兩組字元，專題過程證明了利用Arduino與MAX7219控制大型矩陣的技術可行性，實現了低成本、任何人都可自製的中文字LED看板。透過硬體焊接、程式設計與多次測試，我們獲得了許多經驗並發現了系統設計中的挑戰。

英文摘要

LED display technology is widely used in modern life, however, most of the existing open source materials are designed to display English and use small LED matrix modules. It is still unknown how to create a low-cost, highly editable and portable large-scale LED signage for Chinese users to input Chinese characters to change the font for display and communication. Therefore, the main theme of this project is to explore how to make a low-cost LED matrix and combine Arduino and MAX7219 control module to display Chinese characters in a running light. This study is mainly responsible for the MAX7219 control and the realization of the Chinese running light effect. First of all, we have to learn the basic operation and write the program, and use the small 8x8 LED module to design the dynamic modes such as font drawing, scrolling and blinking, and improve the font design and dynamic display effect through repeated testing. Finally, when applying to a large board, the direction, layout and scrolling time of the font were adjusted again to complete the integration of hardware and software to ensure the stability of the display. Finally, we successfully displayed two sets of characters “慈、醫” and “大、工” on the two LED boards, which proved the technical feasibility of using Arduino and MAX7219 to control the large-scale matrices, and realized a low-cost Chinese character LED signage that can be customized by anyone. Through hardware soldering, programming, and many tests, we gained a lot of experience and discovered the challenges in system design.

前言

LED顯示技術在現代生活中扮演著重要角色，廣泛應用於各種領域，如日常的廣告看板、交通指示燈、以及高科技的顯示設備。相比傳統顯示技術，LED具有高亮度、低能耗、長壽命等優點，這使其成為視覺顯示系統的理想選擇。隨著LED技術的發展，越來越多的應用場景要求LED顯示系統具備動態顯示功能，也就是跑馬燈，尤其是在文字顯示與圖形顯示方面。這次專題主要是在探索如何利用Arduino控制LED顯示內容，實現字體動態顯示的效果，並通過設計與製作16x16 LED顯示板來理解電子硬體設計與程式開發技術。

為了實現這一目標，我們選用了容易操作的MAX7219驅動晶片，該晶片具有多路LED顯示板控制能力，可以有效簡化LED顯示板的驅動與控制。並結合Arduino開發板，設計了由四塊16x16 LED顯示板組成的跑馬燈效果，展示「慈」、「大」、「醫」、「工」四個字。這一設計不僅能夠展示基本的跑馬燈效果，還能實現字體的動態顯示，達到展示文字內容的目的。

在實作過程中，硬體設計與焊接都需要仔細的操作，我們對各個顯示板進行了測試，以確保每塊16x16顯示板能夠正確顯示。程式編寫部分，我們編寫了控制字體顯示與跑馬燈效果的程式碼，並對程式優化去解決顯示延遲與閃爍的問題。特別是在多顯示板同時顯示時，我們發現了不穩定的問題，後來是通過調整程式邏輯與硬體調整，最終實現了穩定且流暢的效果。

這次專題的核心目標，不是只有在於完成一套能夠顯示動態效果的LED顯示裝置，還希望通過這一次積累更多的實作經驗。因為參與這次專題，我不僅加深了對LED顯示技術的理解，還學會了如何將硬體與軟體進行有效的整合，也為未來進一步的研究和應用開發奠定了基礎。

材料與方法

準備Arduino開發板、MAX7219驅動晶片、8x8 LED矩陣模組、電路板、焊接工具

流程圖：

→ 撰寫基礎程式測試LED燈的亮滅控制，熟悉Arduino操作。
→ 設計字型與顯示效果，開發顯示跑馬燈動態效果的程式碼。
→ 使用MAX7219驅動晶片測試8x8 LED矩陣的顯示功能。
→ 焊接LED模組，組成16x16 LED顯示板。
→ 正確接線，確保每塊LED板的電路連接穩定。
→ 將硬體與程式結合，測試16x16 LED顯示板的功能。
→ 調整顯示亮度、字型位置及運行穩定性。
→ 優化跑馬燈效果，讓字型顯示更清晰流暢。
→ 確保每塊板子成功展示「慈、醫」與「大、工」的內容。

方法：
1.小型模擬設計

1-1Arduino 介紹，利用Arduino 的IDE，控制 LED做SOS信號 亮「三短音」、「三長音」、「三短音」。

Arduino 是一個開放式硬體平台，結合易用的硬體和軟體，適合電子開發和互動項目。其核心由開發板（如 Arduino Uno、Nano）和免費的 Arduino IDE 軟體組成，支援使用者以簡單的程式語法控制感測器、LED、馬達等元件。[1]
主要特點：

- 開源性：硬體與軟體均可自由修改與分享。
- 易用性：編程簡單，硬體連接方便，適合新手和專業開發者。
- 多樣性：提供多種開發板，滿足從物聯網到機器人的不同需求。

Led一閃一閃，程式碼可參考[2]

Led做SOS信號

————————————————————

```
void setup() {
  pinMode(8, OUTPUT);}
void loop() {
 // 第一段 for 迴圈：LED 閃爍 3 次，間隔 500 毫秒
 for (int i = 0; i < 3; i++) {
   digitalWrite(8, HIGH); // 亮
   delay(500);          // 等待 500 毫秒
   digitalWrite(8, LOW);  // 暗
   delay(500);          // 等待 500 毫秒 }
 // 第二段 for 迴圈：LED 閃爍 3 次，間隔 1 秒
 for (int i = 0; i < 3; i++) {
   digitalWrite(8, HIGH);
   delay(1000);         // 等待 1 秒
   digitalWrite(8, LOW);
   delay(1000);         // 等待 1 秒 }
}
```

1-2改用8*8 LED板如何接上Arduino，並做一排一排輪播

Matrix Pin	Arduino Pin
1	13
2	12
3	11
4	10
5	16 (Analog 2)
6	17 (Analog 3)
7	18 (Analog 4)
8	19 (Analog 5)

Matrix Pin	Arduino Pin
9	2
10	3
11	4
12	5
13	6
14	7
15	8
16	9

程式碼

————————————————————————

```
// 1-dimensional array of row pin numbers:
const int row[8] = {
  2, 7, 13, 5, 19, 12, 18, 10
};
// 1-dimensional array of column pin numbers:
const int col[8] = {
  6, 17, 16, 3, 11, 4, 8, 9
};

void setup() {
  // Initialize the I/O pins as outputs
  for (int thisPin = 0; thisPin < 8; thisPin++) {
    // Initialize the output pins:
    pinMode(col[thisPin], OUTPUT);
    pinMode(row[thisPin], OUTPUT);
  }

  // Set all COLUMNS to HIGH to ensure all dark
  for (int thisPin = 0; thisPin < 8; thisPin++) {
    digitalWrite(col[thisPin], HIGH);
  }
}

// Function to clear the screen
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() {
  // Display row 3 for 1000 msec
  for(int i = 0; i<8;i++){
  digitalWrite(row[i], HIGH);
   delay(1000);
 // clear screen (all dark)
 clearscreen();
 delay(1000);
 }
 }
```

1-3程式顯示「大」

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);
 }
}
```

1-4研究如何使用Max7219，並在8x8 LED matrix上顯示

接線說明：[3]

1.模塊左邊為輸入端口，右邊為輸出端口。

2.多個模塊串聯時，第1個模塊的輸入端接 CPU，輸出端接第2個模塊的輸入端，第2個模塊的輸出端接第3個模塊的輸入端，以此類推...

Max7219	Arduino Uno
VCC	5V
GND	GND
DIN	PIN
CS	PIN
CLS	PIN

程式碼可參考[4]

1-5用Max7219，程式顯示跑馬燈「大」

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);
// 8x8 矩陣的「大」字圖形，使用二進制表示
byte daCharacter[8] = {
  B00010000,
  B00010000,
  B11111111,
```

```
  B00010000,
  B00101000,
  B01000100,
  B10000010,
  B00000001,
};

void setup() {
  lc.shutdown(0, false);  // 啟動 LED 模組
  lc.setIntensity(0, 8);  // 設定亮度 (0-15)
  lc.clearDisplay(0);     // 清空顯示
}

void loop() {
  // 從左到右逐列顯示「大」
  for (int shift = 7; shift >= 0; shift--) {
    for (int row = 0; row < 8; row++) {
      // 將每列左移 shift 位元，逐步從左至右顯示字元
      byte data = daCharacter[row] << shift;
      lc.setRow(0, row, data);
    }
    delay(250);  // 控制顯示速度
  }
 // 短暫停頓後逐漸從右到左隱去
  delay(500);
  for (int shift = 0; shift < 8; shift++) {
    for (int row = 0; row < 8; row++) {
      // 將每列右移 shift 位元，逐步從右至左隱去字元
      byte data = daCharacter[row] >> shift;
      lc.setRow(0, row, data);
    }
    delay(250);  // 控制隱去速度
```

```
}

// 清除顯示後，暫停一下，準備下一次顯示
lc.clearDisplay(0);
delay(1000);}
```

1-6串聯8x8 LED matrix， 在兩個8x8 LED matrix上顯示「慈」下半部

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, // 額外的空行，保持 16 行
  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
}

  // 顯示第一個板子上的字形（固定顯示）
  for (int i = 0; i < 8; i++) {
    lc.setRow(0, i, daCharacter[i]);  // 在第一個板子顯示字形（固定）
  }
}

void loop() {
  // 讓第二個板子上的字形從左到右滾動
  for (int offset = 0; offset < 16; offset++) {
```

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

  // 顯示第二個板子上的字形，從左到右滾動
  for (int row = 0; row < 8; row++) {
    if (offset + row < 16) {
      lc.setRow(1, row, daCharacter[offset + row]);  // 只顯示第二個板子上的字形，滾動其部分
    }
  }

  delay(200); // 控制滾動速度
}
}
```

1-7將字分成上下部分，分別顯示做成跑馬燈

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
    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() {
  // 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}
```

1-8 焊接並組裝LED燈，完成4塊16x16 LED板的硬體設計。

1-9在兩個板子上分別顯示『「慈」、「醫」』、『「大」、「工」』，下方程式碼為『「大」、「工」』。

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 (原來的字形)
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 (新的字形)
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}
```

1-10 透過Arduino程式控制，讓16x16 LED板分別顯示「慈」、「大」、「醫」、「工」四個字。

結果與討論

這次專題，我們成功完成了兩塊16x16 LED顯示板，讓它們分別呈現「慈、醫」和「大、工」兩組字元並做成跑馬燈。硬體部分是要把LED模組焊接到電路板上，確定接線正確而且能穩定供電；軟體部分是透過Arduino IDE撰寫程式，讓我們要的字成功在LED上顯示。

雖然原本的計畫是要做四塊顯示板分別顯示「慈、大、醫、工」四個字，但因為時間和技術上的限制，我們最後只完成了兩塊。但是，我們也因為臨時的調整，讓單個Arduino控制板可以同時控制兩個字元的顯示，學到了如何用一個Arduino的控制LED上的兩個字。

過程中有遇到不少挑戰，像是焊接時正確性要求度很高，焊錯就會導致電路無法正常運作，我們經過反覆的測試並修正順利完成。程式設計方面，剛開始效果有點瑕疵，但透過持續的修改，最後達成較穩定且順暢的效果。

總的來說，這次專題讓我們學到了Arduino多模組的控制，以及LED矩陣和驅動晶片的應用。同時，我們也在過程中學會如何應對臨時的變化，並調整計畫。

參考文獻

1.《Arduino入門》第一篇：認識Arduino

[Internet] Nedlands (TW): 傑森創工；2020 Jan. 9

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

2.《Arduino入門》第三篇：用程式控制LED

[Internet] Nedlands (TW): 傑森創工；2020 Jan. 15

Available from: https://blog.jmaker.com.tw/arduino-tutorials-3/

3.《Ray的Arduino教學》使用MAX7219模組顯示

[Internet] Nedlands (TW): RayArduino

Available from: https://sites.google.com/view/rayarduino/led-max-7219

4.《Arduino進階教學課程》MAX7219驅動點陣模組LED Matrix動畫顯示

[Internet] Nedlands (TW): 米羅科技有限公司；Dec. 28

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