臺北市立大安高級工業職業學校
電子科
專題報告
16*16彩色貪吃蛇遊戲機
16*16 COLOR GREEDY SNAKE GAME CONSOLE

學生　組長：吳晉緯
組員：李汪
組員：周禕
組員：林哲宇

指導老師：黃建中老師

2022年1月10日

目錄

第一章 前言

1-1專題製作背景及目的

- 貪吃蛇是以前經典的手機遊戲，是跨時代的經典，而我們現在就有機會可以將它用16 x 16 RGB LED點矩陣的硬件表現出來，也是別有一番風味的。
- 專題的題目我們選擇貪吃蛇，是因為難度適中，非常適合當作專題，且這東西具有基本的娛樂性，可以無聊的時候用來打發時間，可謂是一舉兩得。

1-2預期成果

- 1. 可以供兩個同學遊玩貪吃蛇，且可以自己挑選蛇的顏色，可以互相對戰
- 2. 可以顯示7種顏色的像素
- 3. 液晶顯示器可以顯示分數或其它資訊
- 4. 外觀精美
- 5. 用兩顆鋰電池就可以正常使用
- 6. 蜂鳴器可以發出音效(後來沒加)

第二章 理論探討

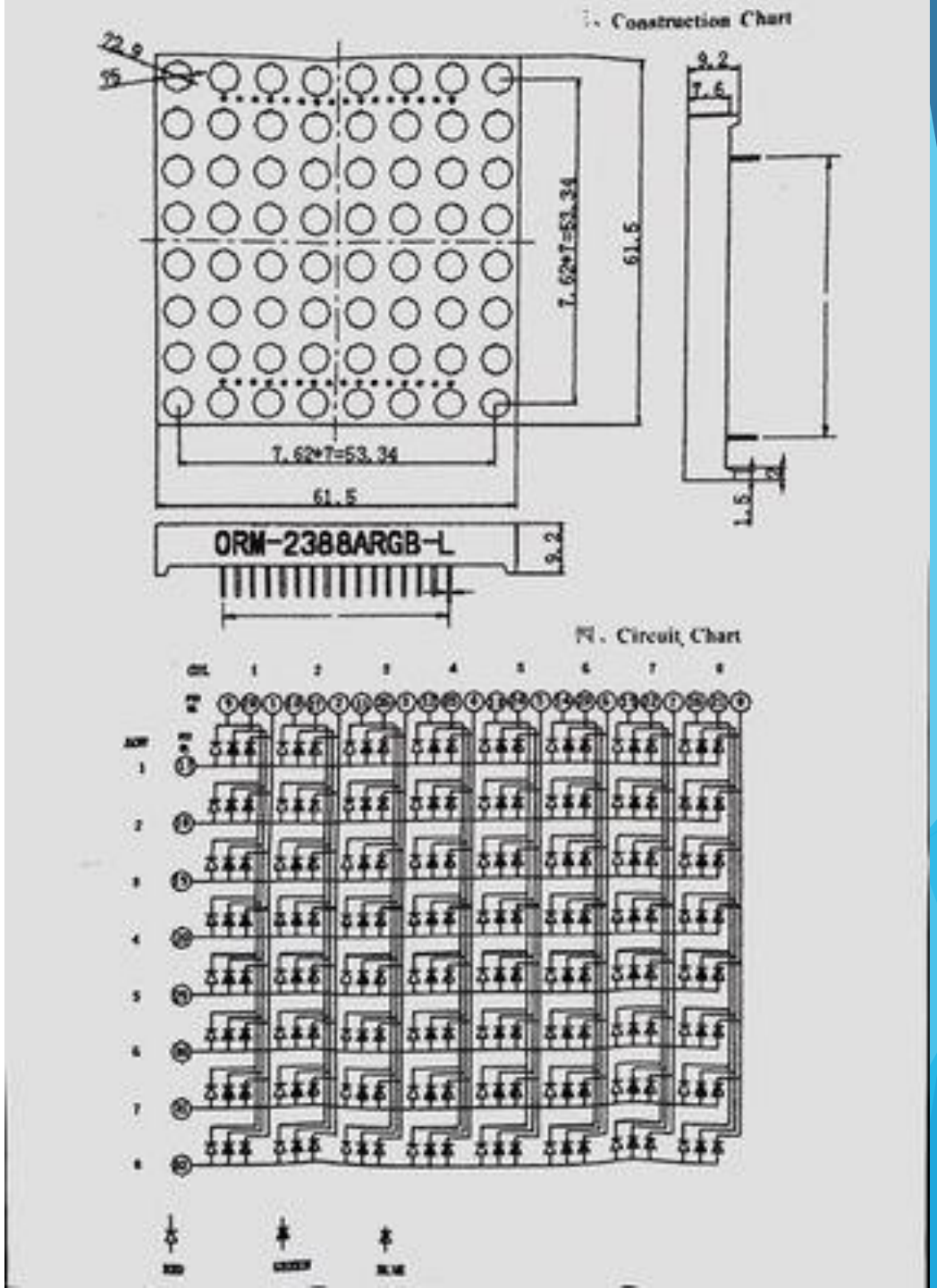
2-1電路設計

2-1-1 Altium Designer

Altium designer是altium公司開發的一款電子設計自動化軟體，用於原理圖、PCB、FPGA設計。 結合了板級設計與FPGA設計。

2-1-2　元件介紹

RGB 點矩陣



P通道MOSFET

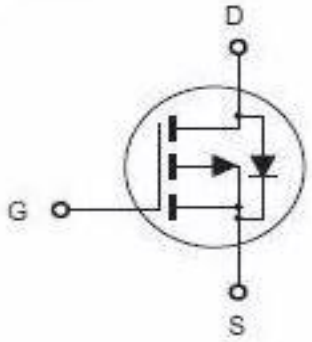
FEATURES

- High dense cell design for extremely low $R_{DS(ON)}$
- Rugged and reliable
- Case Material: Molded Plastic.

SI2301
P-Channel MOSFET

1.Gate
2.Source
3.Drain

SOT-23



Absolute Maximum Ratings (TA=25°C, unless otherwise noted)

Parameter	Symbol	Ratings	Units
Drain-Source Voltage	VDS	-20	V
Gate-Source Voltage	VGS	±8	V
Drain Current (Continuous)	ID	-2.8	A
Drain Current (Pulsed) [1]	IDM	-10	A
Total Power Dissipation @TA=25oC	PD	1.25	W
Operating Junction and Storage Temperature Range	T_j, T_{stg}	-55 to +150	°C
Thermal Resistance Junction to Ambient (PCB mounted)[2]	$R_{\theta JA}$	100	°C/W

Electrical Characteristics (TA=25°C, unless otherwise noted)

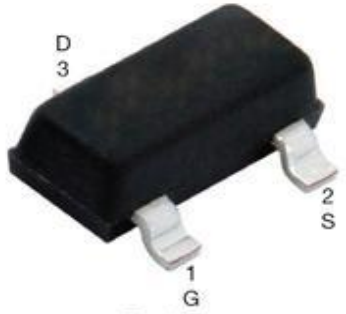
N通道MOSFET

VISHAY
www.vishay.com

2N7002K
Vishay Siliconix

N-Channel 60 V (D-S) MOSFET

SOT-23 (TO-236)



Top View

Marking code: 7K

PRODUCT SUMMARY	
V_{DS} (V)	60
$R_{DS(on)}$ max. (Ω) at V_{GS} = 10 V	2
Q_g typ. (nC)	0.4
I_D (mA)	300
Configuration	Single

FEATURES

- Low on-resistance: 2 Ω
- Low threshold: 2 V (typ.)
- Low input capacitance: 25 pF
- Fast switching speed: 25 ns
- Low input and output leakage
- TrenchFET® power MOSFET
- 2000 V ESD protection
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

Pb-free Available
RoHS* Available
HALOGEN FREE Available

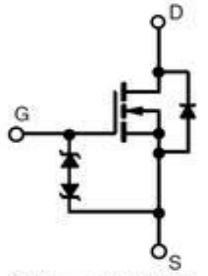
Note
* This datasheet provides information about parts that are RoHS-compliant and / or parts that are non RoHS-compliant. For example, parts with lead (Pb) terminations are not RoHS-compliant. Please see the information / tables in this datasheet for details

BENEFITS

- Low offset voltage
- Low voltage operation
- Easily driven without buffer
- High speed circuits
- Low error voltage

APPLICATIONS

- Direct logic-level interface: TTL/CMOS
- Drivers:
  relays, solenoids, lamps, hammers, display, memories, transistors, etc.
- Battery operated systems
- Solid state relays



N-Channel MOSFET

穩壓IC

Product Folder | Order Now | Technical Documents | Tools & Software | Support & Community | Reference Design

TEXAS INSTRUMENTS

LM78M05-MIL Series 3-Terminal 500-mA Positive Voltage Regulator

1 Features

- Output Current in Excess of 0.5 A
- No External Components
- Internal Thermal Overload Protection
- Internal Short Circuit Current-Limiting
- Output Transistor Safe-Area Compensation
- Available in 3-Pin TO-220, TO-252, and TO packages
- Output Voltage: 5 V

2 Applications

- Electronic Point-of-Sale
- Medical and Health Fitness Applications
- Printers
- Appliances and White Goods
- TVs and Set-Top Boxes

3 Description

The LM78M05-MIL three-pin positive voltage regulator employs built-in current limiting, thermal shutdown, and safe-operating area protection, which makes them virtually immune to damage from output overloads.

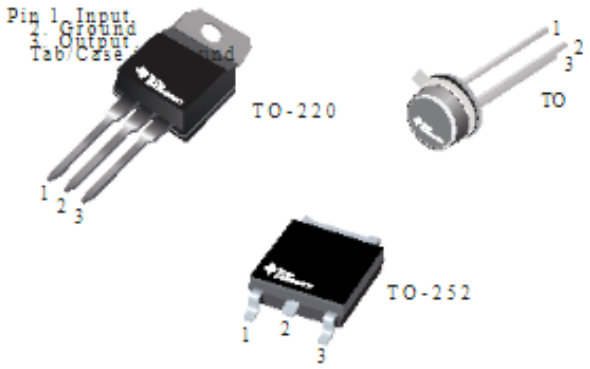
With adequate heat sinking, they can deliver in excess of 0.5-A output current. Typical applications would include local (on-card) regulators which can eliminate the noise and degraded performance associated with single-point regulation.

Device Information(1)

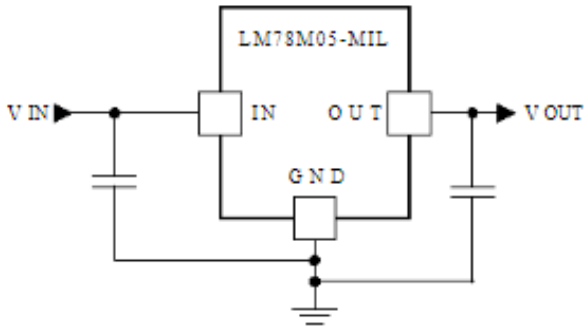
PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM78M05	TO-220 (3)	10.16 mm × 14.986 mm
	TO-252 (3)	6.10 mm × 6.58 mm
	TO (3)	9.14 mm × 9.14 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

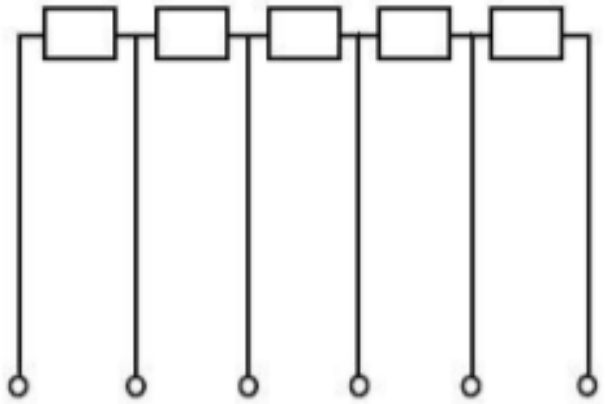
Available Packages

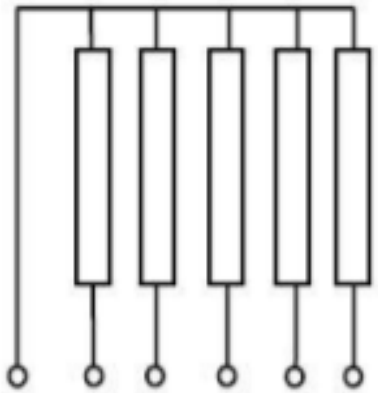


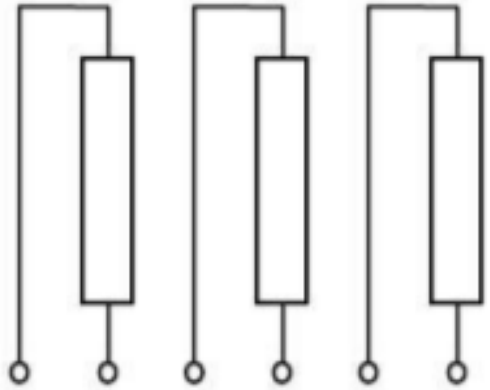
Simplified Application



排阻







74138解碼器

SN74LS138

1-of-8 Decoder/ Demultiplexer

The LSTTL / MSI SN74LS138 is a high speed 1-of-8 Decoder / Demultiplexer. This device is ideally suited for high speed bipolar memory chip select address decoding. The multiple input enables allow parallel expansion to a 1-of-24 decoder using just three LS138 devices or to a 1-of-32 decoder using four LS138s and one inverter. The LS138 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all ON Semiconductor TTL families.

- Demultiplexing Capability
- Multiple Input Enable for Easy Expansion
- Typical Power Dissipation of 32 mW
- Active Low Mutually Exclusive Outputs
- Input Clamp Diodes Limit High Speed Termination Effects

GUARANTEED OPERATING RANGES

Symbol	Parameter	Min	Typ	Max	Unit
V CC	Supply Voltage	4.75	5.0	5.25	V
TA	Operating Ambient Temperature Range	0	25	70	°C
I OH	Output Current – High			– 0.4	mA
I OL	Output Current – Low			8.0	mA

ON Semiconductor
Formerly a Division of Motorola
http://onsemi.com

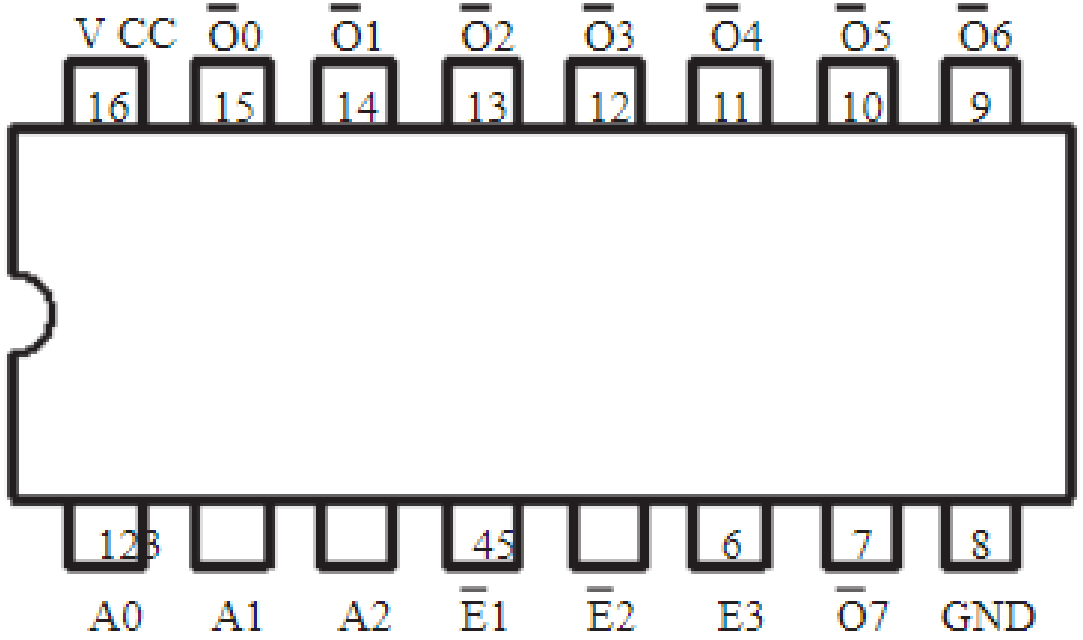
LOW
POWER
SCHOTTKY

PLASTIC
N SUFFIX
CASE 648

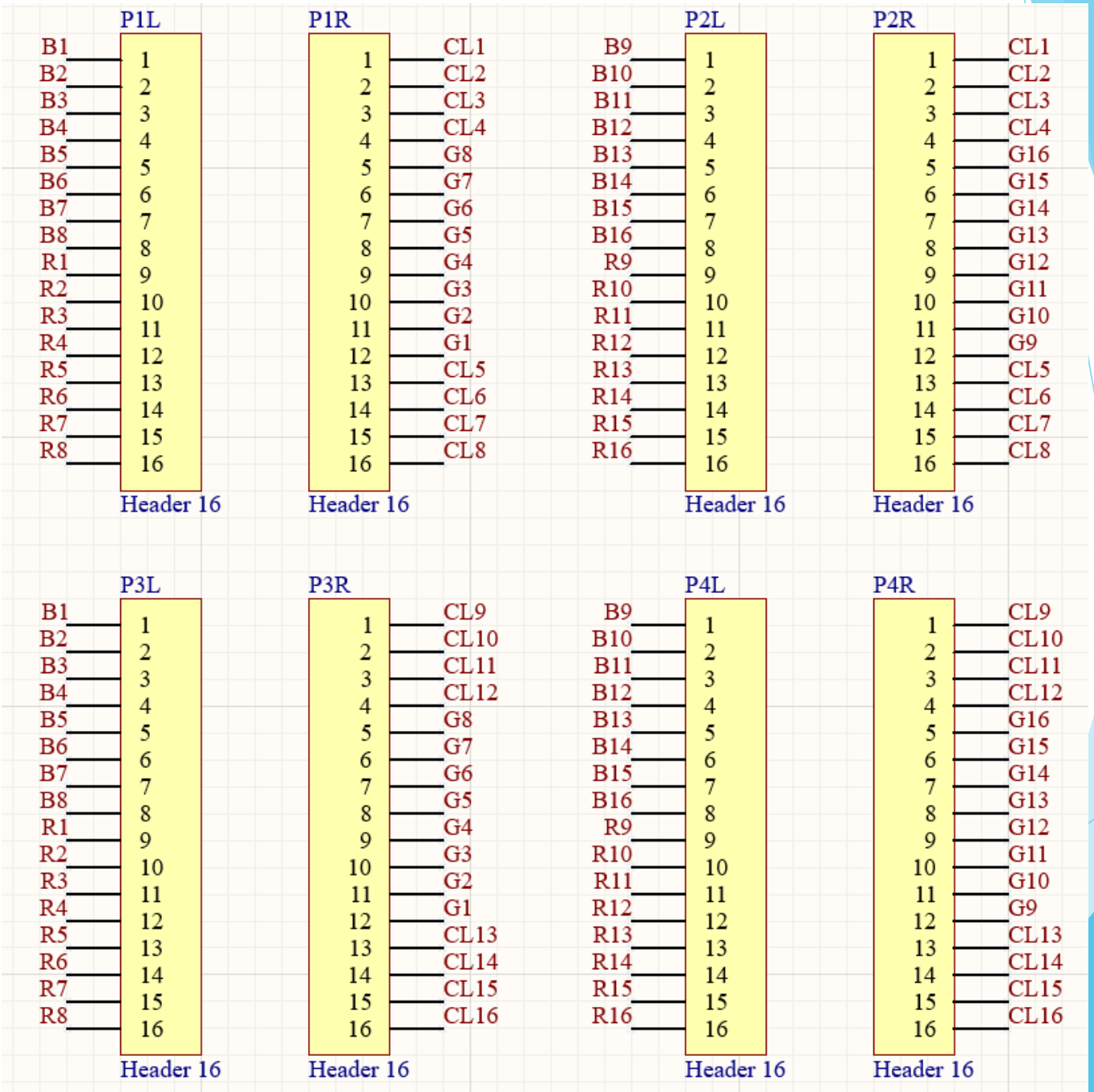
SOIC
D SUFFIX
CASE 751B

SN74LS138

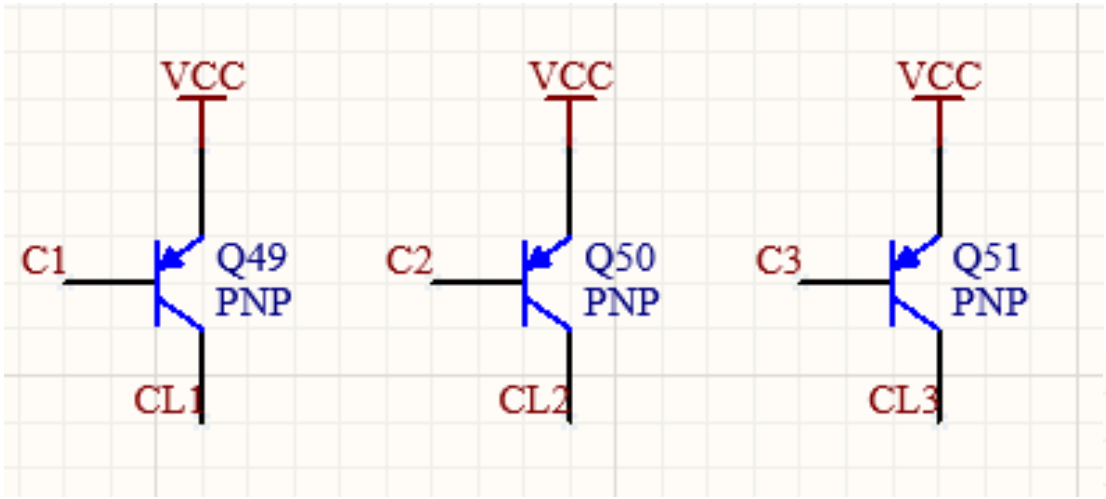
CONNECTION DIAGRAM DIP (TOP VIEW)

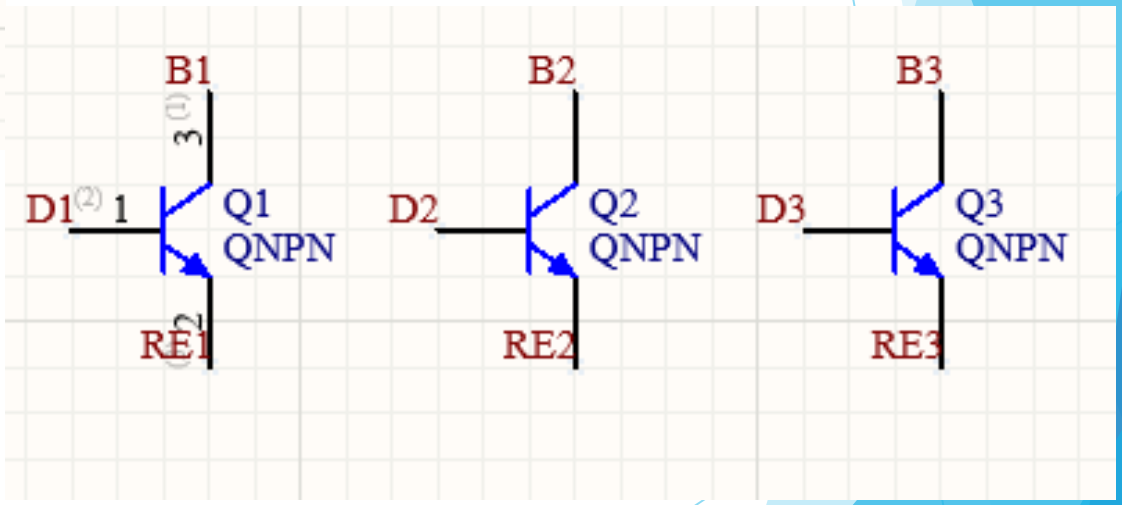


RGB點矩陣

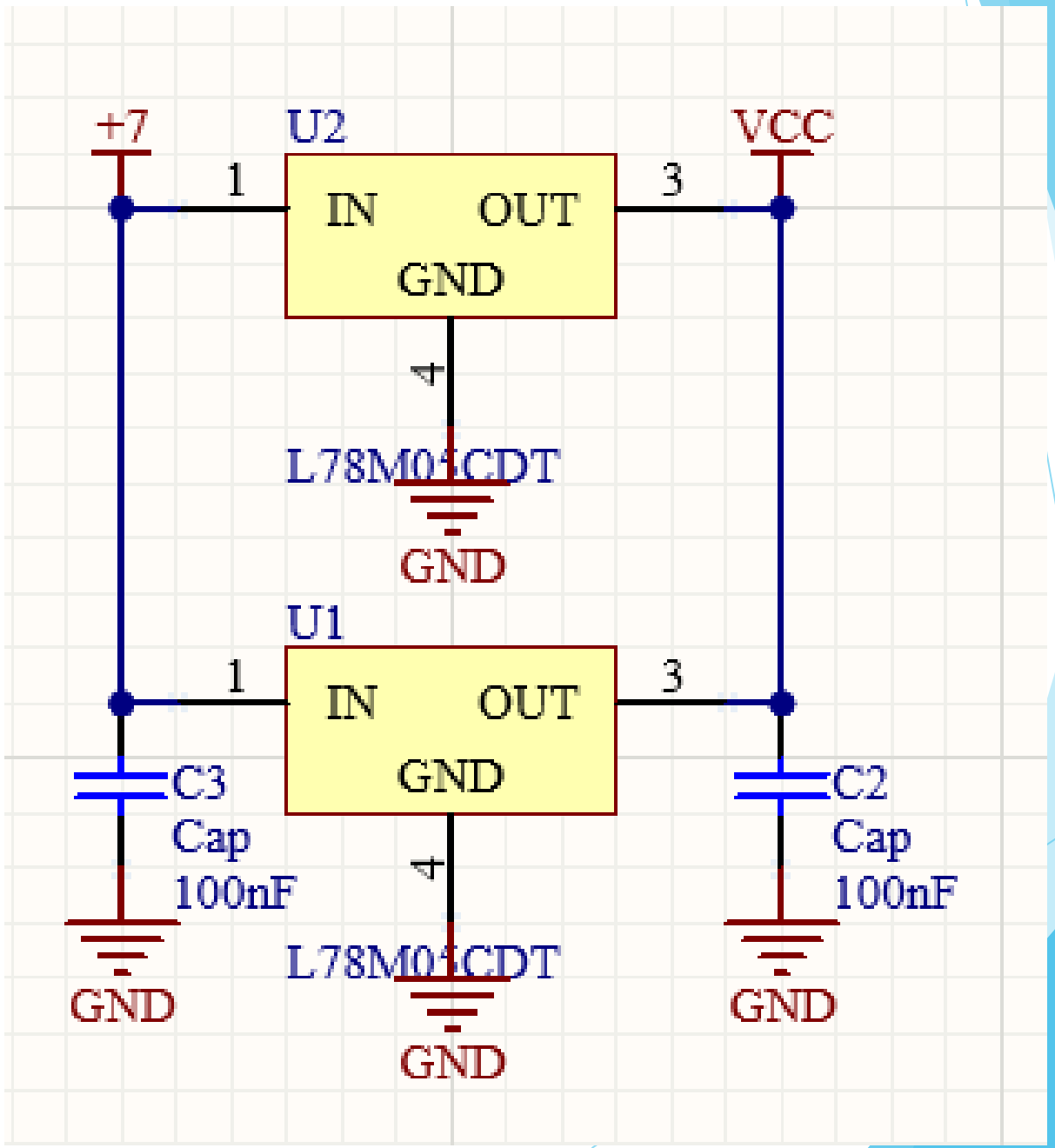

P1L
B1 1
B2 2
B3 3
B4 4
B5 5
B6 6
B7 7
B8 8
R1 9
R2 10
R3 11
R4 12
R5 13
R6 14
R7 15
R8 16
Header 16
P1R
1 CL1
2 CL2
3 CL3
4 CL4
5 G8
6 G7
7 G6
8 G5
9 G4
10 G3
11 G2
12 G1
13 CL5
14 CL6
15 CL7
16 CL8
Header 16
P2L
B9 1
B10 2
B11 3
B12 4
B13 5
B14 6
B15 7
B16 8
R9 9
R10 10
R11 11
R12 12
R13 13
R14 14
R15 15
R16 16
Header 16
P2R
1 CL1
2 CL2
3 CL3
4 CL4
5 G16
6 G15
7 G14
8 G13
9 G12
10 G11
11 G10
12 G9
13 CL5
14 CL6
15 CL7
16 CL8
Header 16
P3L
B1 1
B2 2
B3 3
B4 4
B5 5
B6 6
B7 7
B8 8
R1 9
R2 10
R3 11
R4 12
R5 13
R6 14
R7 15
R8 16
Header 16
P3R
1 CL9
2 CL10
3 CL11
4 CL12
5 G8
6 G7
7 G6
8 G5
9 G4
10 G3
11 G2
12 G1
13 CL13
14 CL14
15 CL15
16 CL16
Header 16
P4L
B9 1
B10 2
B11 3
B12 4
B13 5
B14 6
B15 7
B16 8
R9 9
R10 10
R11 11
R12 12
R13 13
R14 14
R15 15
R16 16
Header 16
P4R
1 CL9
2 CL10
3 CL11
4 CL12
5 G16
6 G15
7 G14
8 G13
9 G12
10 G11
11 G10
12 G9
13 CL13
14 CL14
15 CL15
16 CL16
Header 16

驅動電路

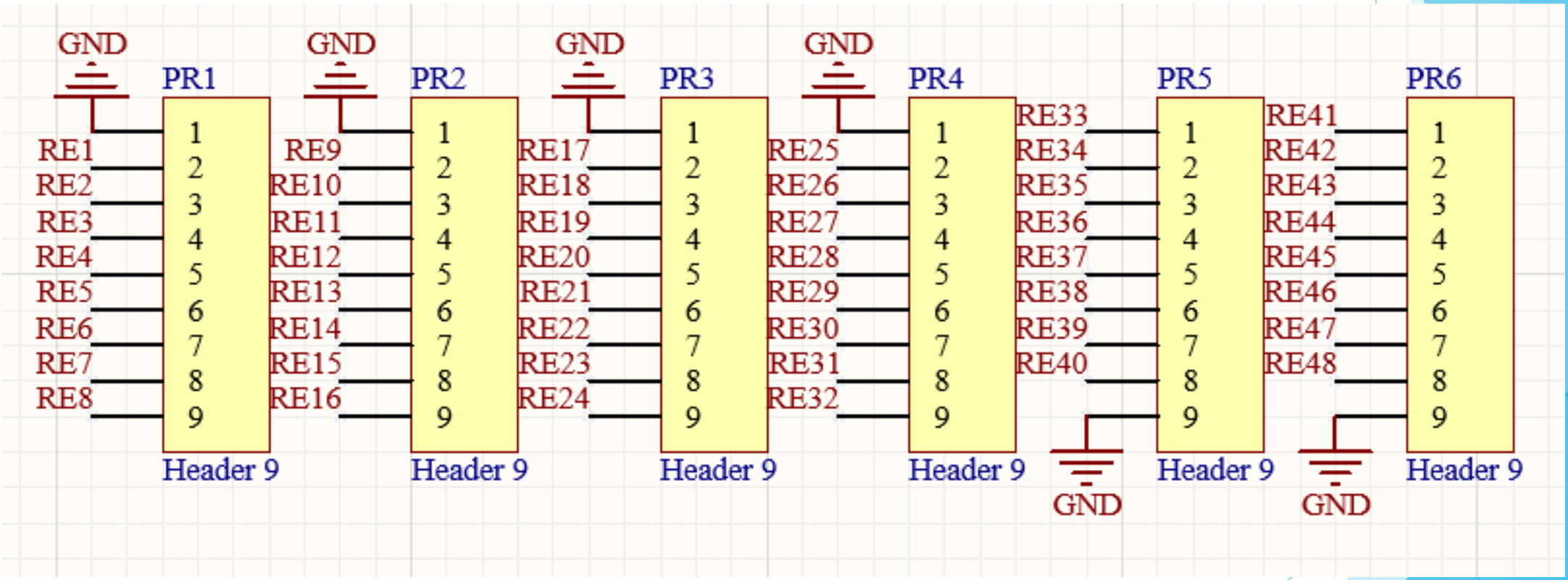

VCC
VCC
VCC
C1
Q49
PNP
C2
Q50
PNP
C3
Q51
PNP
CL1
CL2
CL3


B1
B2
B3
3
D1
1
Q1
QNPN
D2
Q2
QNPN
D3
Q3
QNPN
RE1
RE2
RE3

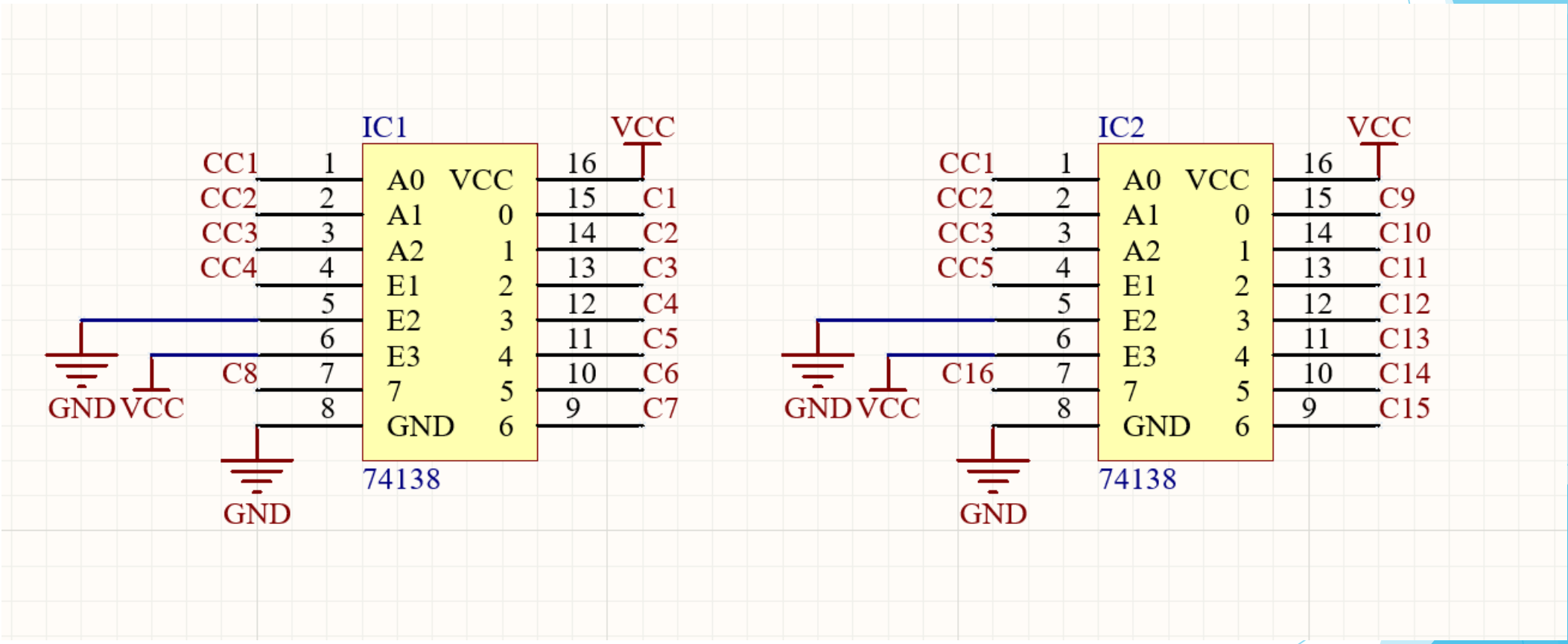
穩壓IC


+7
U2
VCC
1
IN
OUT
3
GND
4
L78M05CDT
GND
U1
1
IN
OUT
3
GND
C3
Cap
100nF
C2
Cap
100nF
4
L78M05CDT
GND
GND
GND

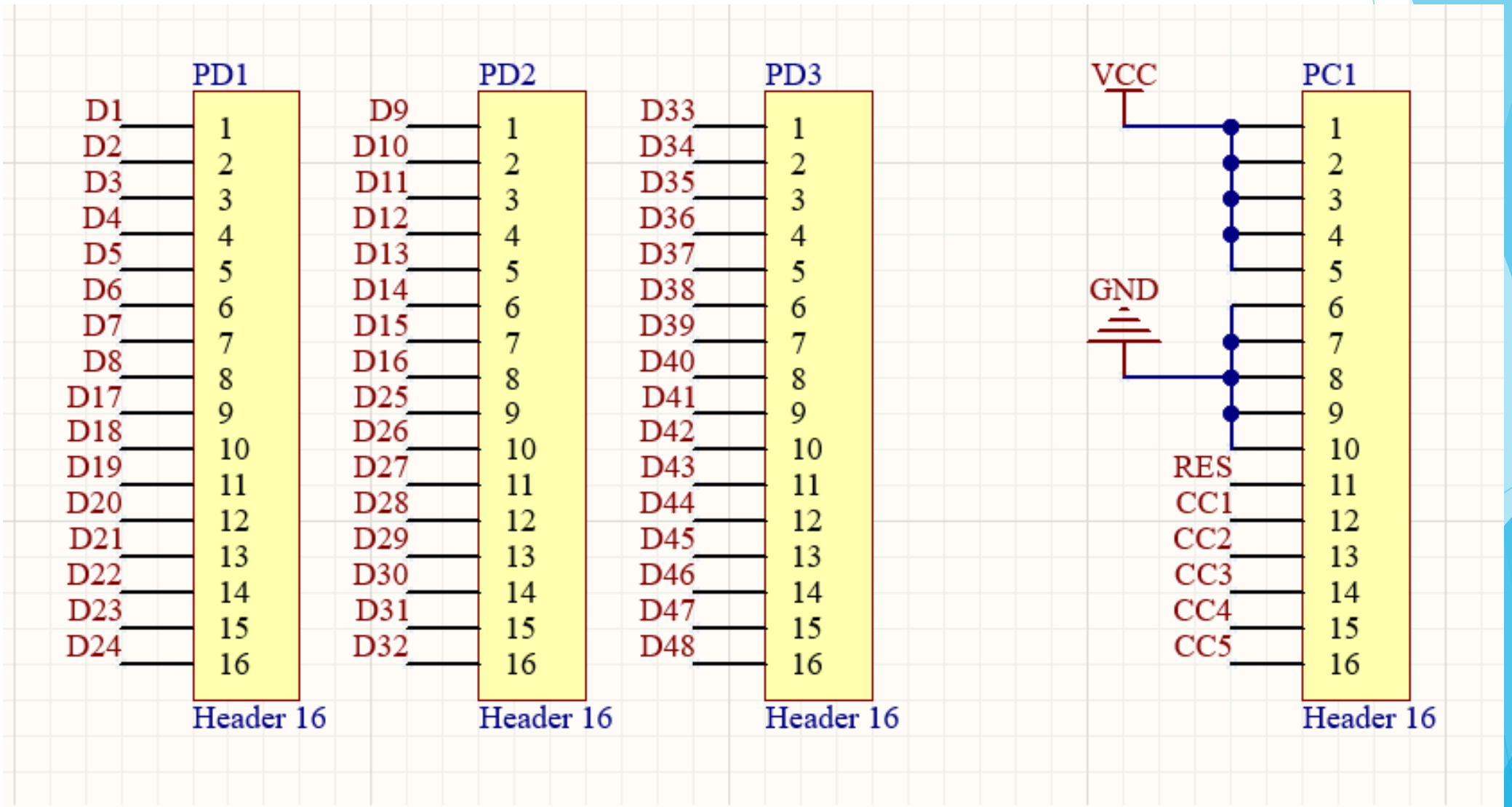
排阻


GND
GND
GND
GND
PR1
PR2
PR3
PR4
PR5
PR6
RE1
RE2
RE3
RE4
RE5
RE6
RE7
RE8
RE9
RE10
RE11
RE12
RE13
RE14
RE15
RE16
RE17
RE18
RE19
RE20
RE21
RE22
RE23
RE24
RE25
RE26
RE27
RE28
RE29
RE30
RE31
RE32
RE33
RE34
RE35
RE36
RE37
RE38
RE39
RE40
RE41
RE42
RE43
RE44
RE45
RE46
RE47
RE48
1
2
3
4
5
6
7
8
9
Header 9
Header 9
Header 9
Header 9
Header 9
Header 9
GND
GND

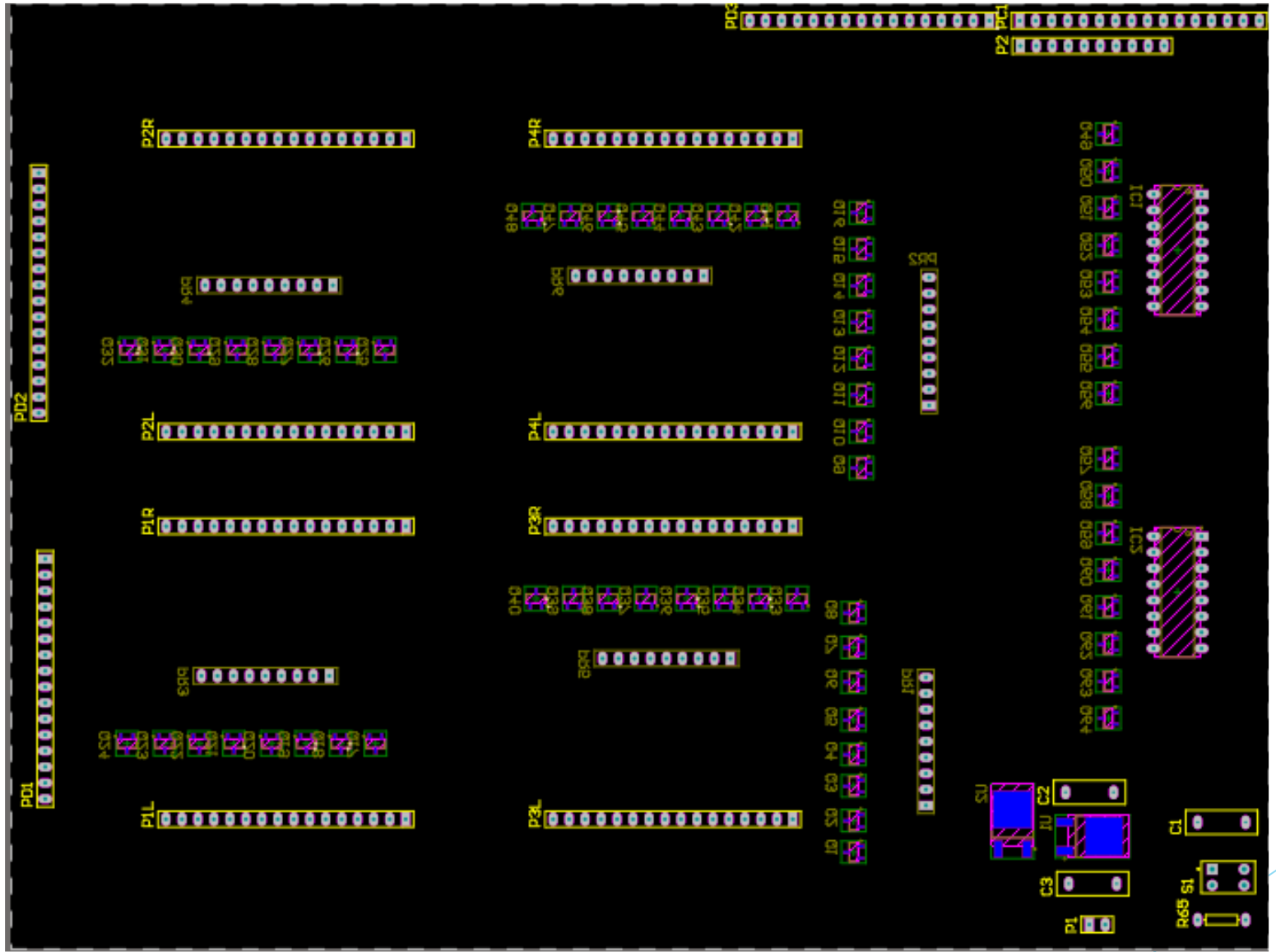
解碼電路


IC1
VCC
CC1
CC2
CC3
CC4
C8
GND
VCC
GND
A0
A1
A2
E1
E2
E3
7
GND
VCC
0
1
2
3
4
5
6
7
74138
C1
C2
C3
C4
C5
C6
C7
IC2
VCC
CC1
CC2
CC3
CC5
C16
GND
VCC
GND
74138
C9
C10
C11
C12
C13
C14
C15

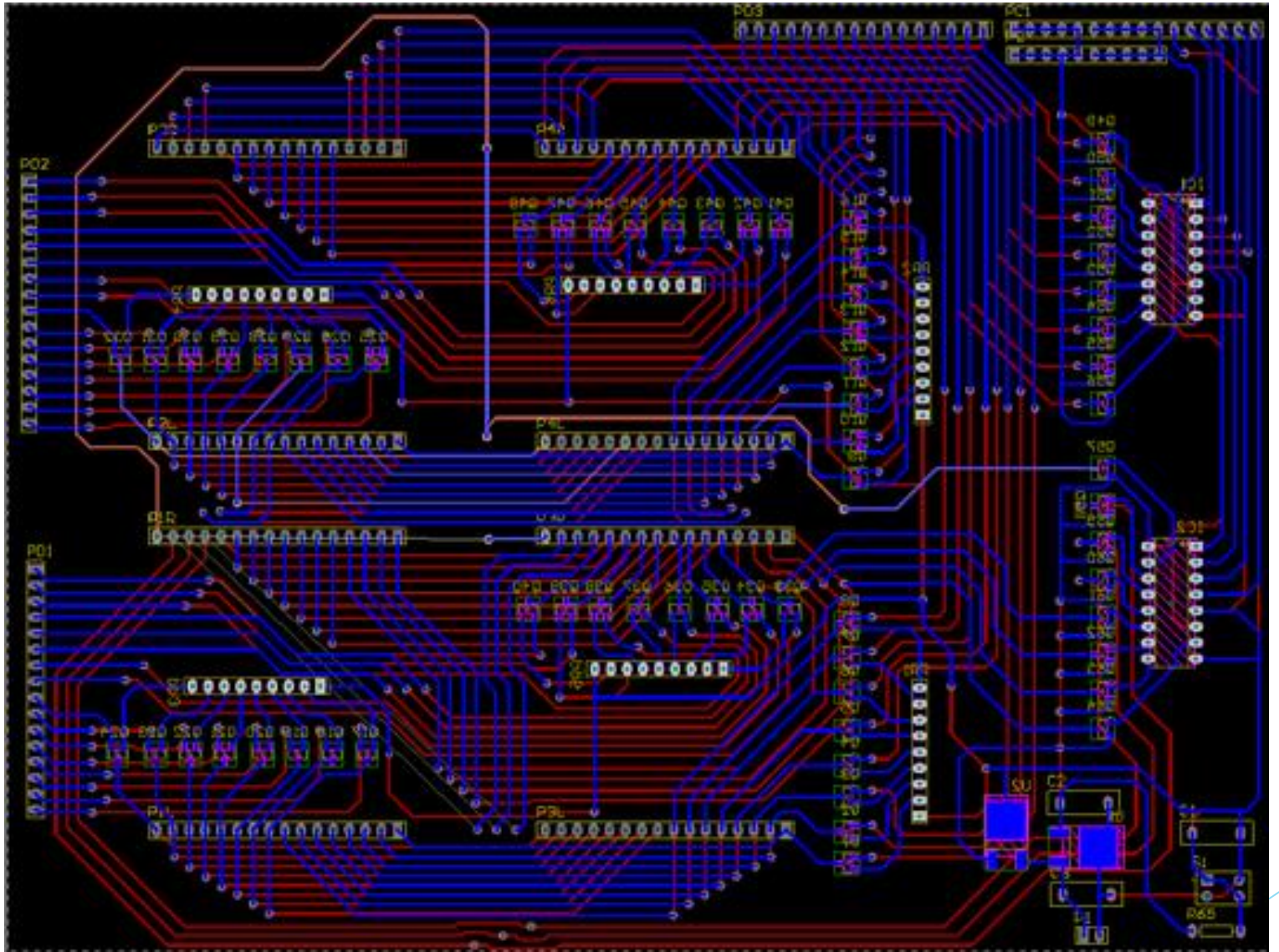
資料線和控制線


PD1
D1
D2
D3
D4
D5
D6
D7
D8
D17
D18
D19
D20
D21
D22
D23
D24
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Header 16
PD2
D9
D10
D11
D12
D13
D14
D15
D16
D25
D26
D27
D28
D29
D30
D31
D32
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Header 16
PD3
D33
D34
D35
D36
D37
D38
D39
D40
D41
D42
D43
D44
D45
D46
D47
D48
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Header 16
VCC
GND
PC1
RES
CC1
CC2
CC3
CC4
CC5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Header 16

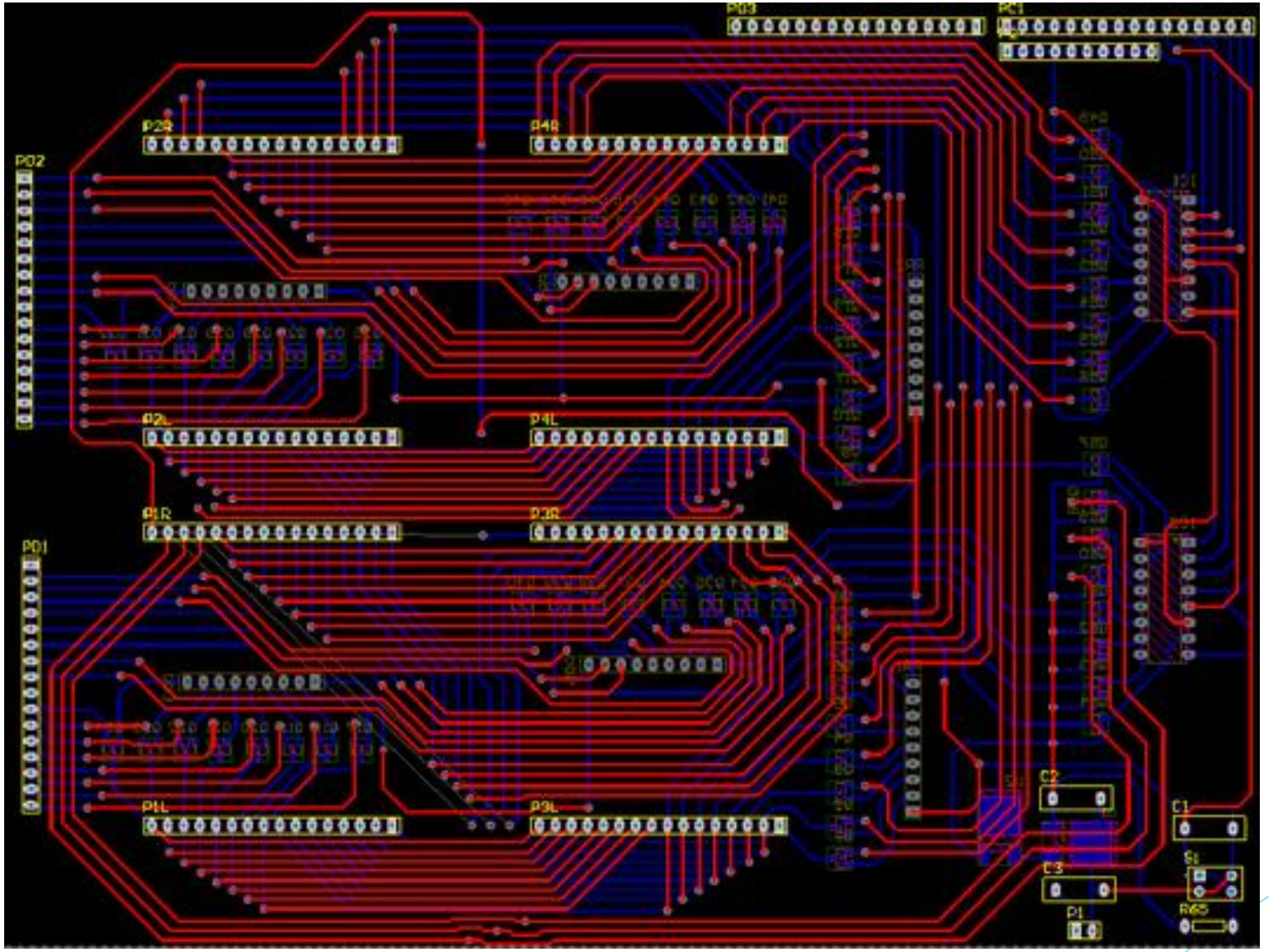
電路板佈線(元件擺放位置)



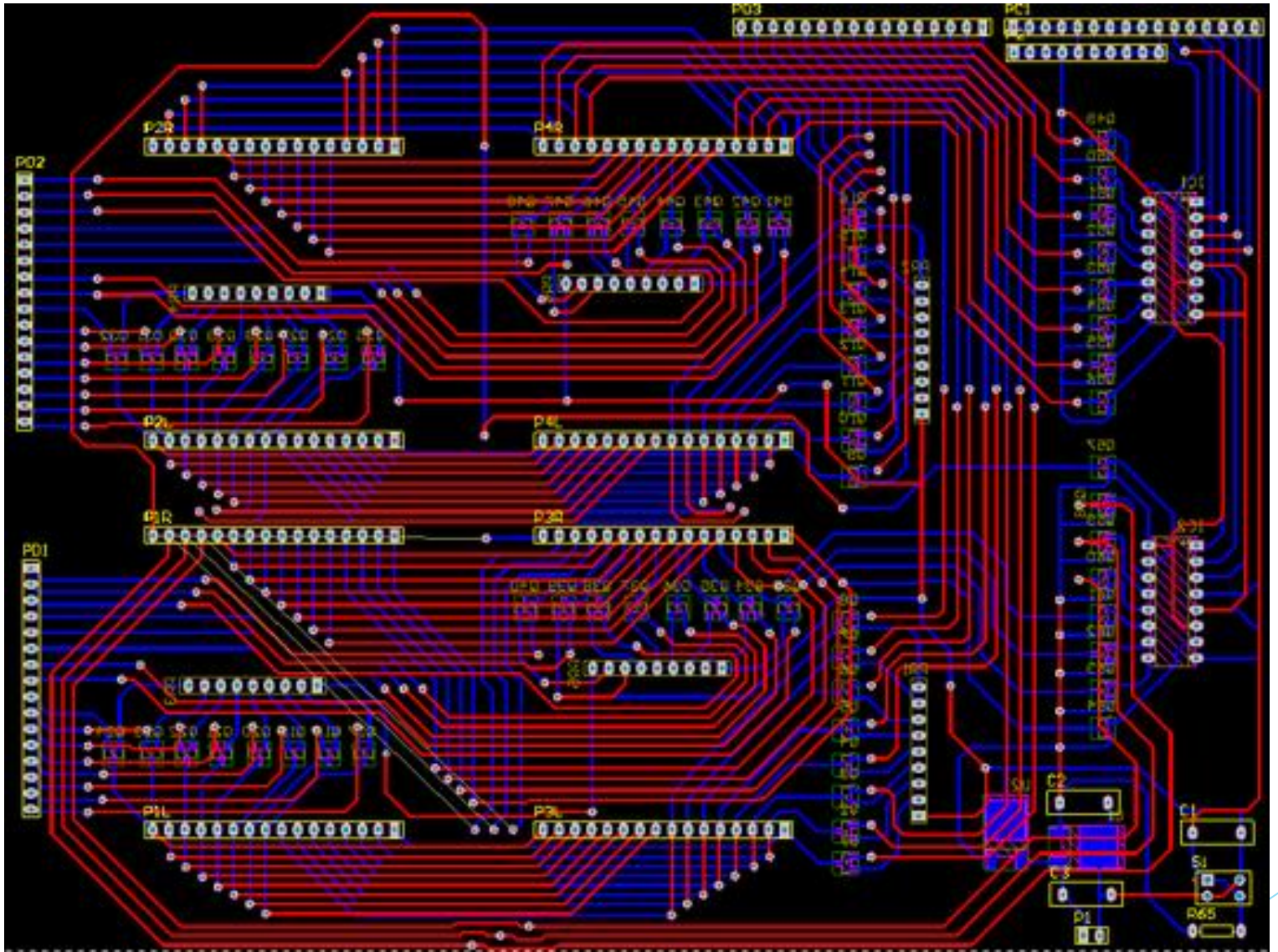
電路板佈線(bottom layer)



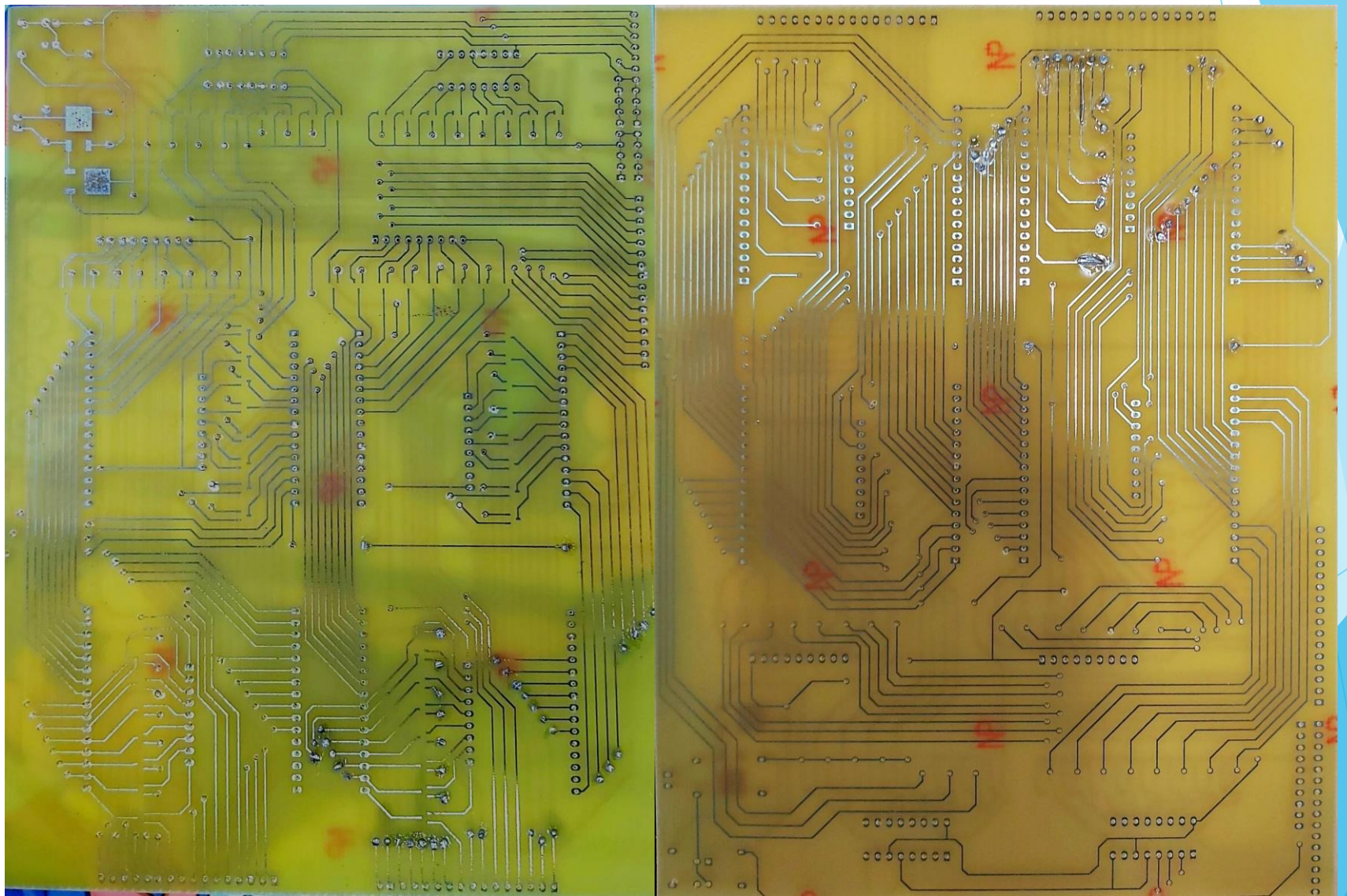
電路板佈線(top layer)



電路板佈線



電路板成品



2-2軟體設計

2-2-1 Arduino IDE

Arduino系列電路板的設計大多使用Atmel AVR 單片機。這些電路板配有一組數位和類比I/O引腳，可以連接各種擴充板或麵包板和其他電路。這些電路板具有串列埠，包括某些型號上的通用串列匯流排（USB），也用於從個人電腦載入程式。

2-2-2 Arduino Mega2560

在Arduino的Mega2560是一個基於微控制器板ATmega2560。它有 54 個數位輸入/輸出引腳（其中 15 個可用作 PWM 輸出）、16 個類比輸入、4 個 UART（硬件串列通訊埠）、一個 16 MHz 晶體振盪器、一個 USB 接頭、一個電源插孔、一個 ICSP 接頭、和一個重置按鈕。

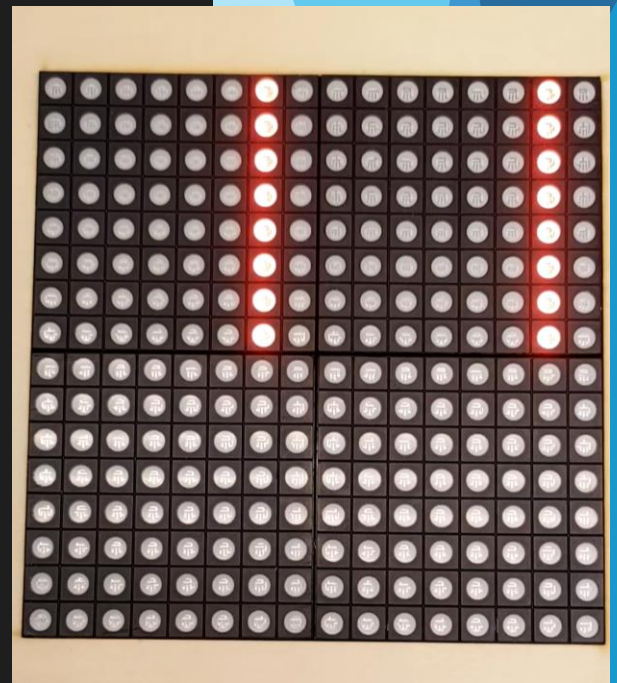
2-2-3 Visual Studio Code

Visual Studio Code（簡稱 VS Code）是一款由微軟開發且跨平台的免費原始碼編輯器。該軟體支援語法突顯、程式碼自動補全（又稱 IntelliSense）、程式碼重構功能，並且內建了命令列工具和 Git 版本控制系統，使用者可以更改佈景主題和鍵盤捷徑實現個人化設定。

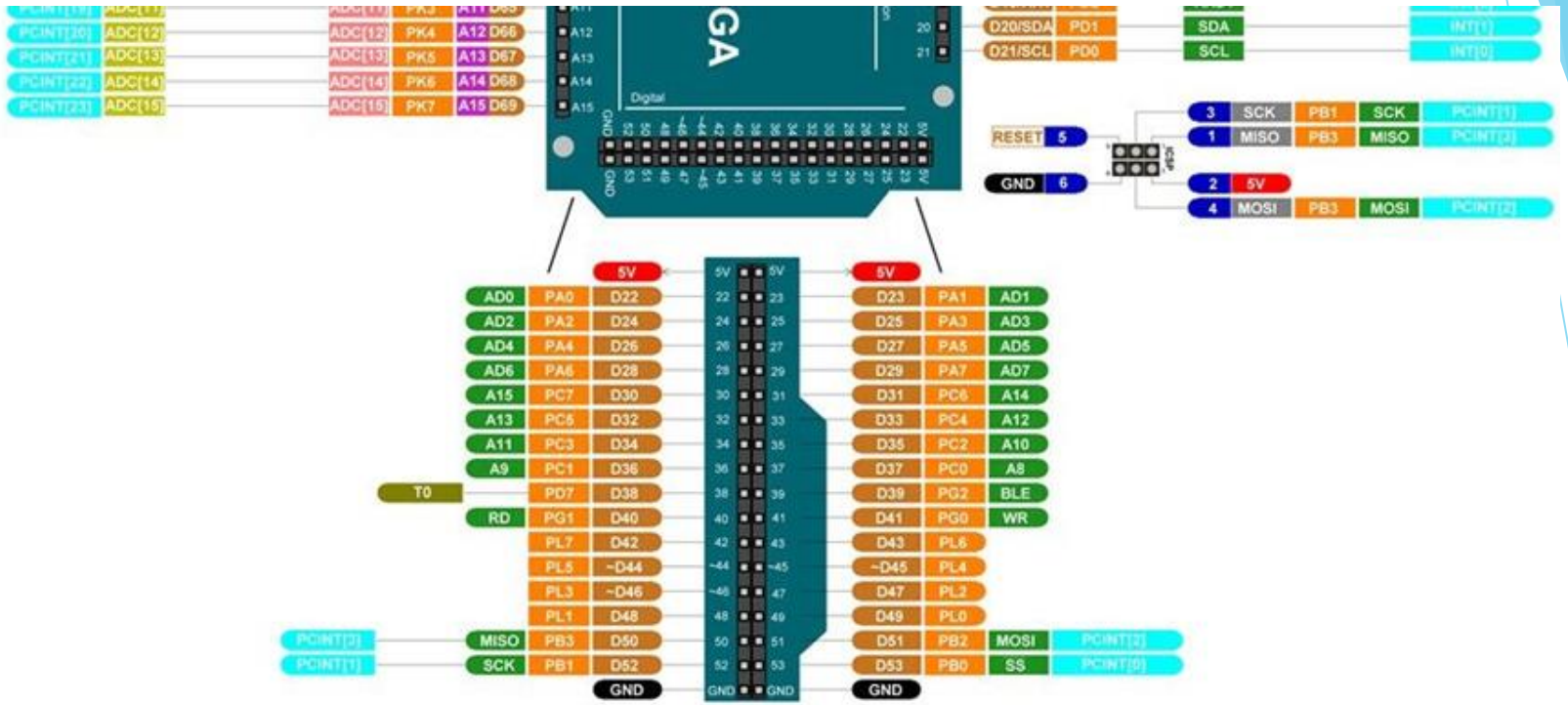

isVideo
isUrl
isElement
isObject
// Check if boxer is already active
if ($("#boxer").length > 0 || isImage || isVideo || isUrl
return;
}
// Kill event
_killEvent(e);
// Cache internal data
data = $.extend({}, {
$window: $(window),
$body: $("body"),
$target: $target,
$object: $object,
visible: false,
resizeTimer: null,
touchTimer: null,
gallery: {
active: false
軟體

1-1. 矩陣掃描

```
ISR(TIMER1_OVF_vect)
{
    TCNT1 = -5;                     //5 個 16K 後產生中斷
    static byte scan_bcd = 0;       //設定掃描計數
    scan_bcd = (scan_bcd + 1) % 16; //掃描計數+1(數值為 1~16)
    Binary(scan_bcd);               //轉二進位並存值

    clear_led(); //掃描及資料線腳位暫存器紀錄歸零

    byte S_LED = 0; //暫存像素資料

    for (byte ii_data = 0; ii_data < 16; ii_data++) //掃描 16 顆 LED(每顆包括紅綠藍)
    {
        if (scan_bcd < 8) //如果目前掃描的是前 8 條(掃描上面兩個點矩陣)
        {
            if (ii_data < 8) //前 8 顆 LED 資料的取得方法
                S_LED = Matrix[7 - ii_data][scan_bcd];
            else //後 8 顆 LED 資料的取得方法
                S_LED = Matrix[15 - ii_data][scan_bcd + 8];
        }
        else //如果目前掃描的是後 8 條(掃描下面兩個點矩陣)
        {
            if (ii_data < 8) //前 8 顆 LED 資料的取得方法
                S_LED = Matrix[15 - ii_data][scan_bcd - 8];
            else //後 8 顆 LED 資料的取得方法
                S_LED = Matrix[23 - ii_data][scan_bcd];
        }

        if (S_LED % 2 == 1)             //LSB = 1
            RED_DigitalWrite(ii_data); //儲存紅色
        S_LED /= 2;
        if (S_LED % 2 == 1)
            GREEN_DigitalWrite(ii_data); //儲存綠色
        S_LED /= 2;
        if (S_LED % 2 == 1)              //MSB = 1
            BLUE_DigitalWrite(ii_data); //儲存藍色
    }

    PORTJ |= 0xFF; //掃描線關閉

    equal(); //資料線送出

    for (byte cc = 0; cc < 3; cc++) //送出掃描線(CBA)
    {
        if (scan_binary_array[cc + 1] == 1) //如果內容是 1 就送出
            PORTE |= _BV(scan_pin[cc]);
    }

    if (scan_binary_array[0] == 0) //送出掃描線(選擇 7138 解碼器，分別控制上下兩顆矩陣)
        PORTJ &= ~0x02;             //選擇上面的矩陣
    else
        PORTJ &= ~0x01; //選擇下面的矩陣
}

void RED_DigitalWrite(byte pin) //紅色資料線的紀錄
{
    if (pin < 8) //前八顆紅色紀錄
    {
        PORT_A |= _BV(pin);
    }
    else //後八顆紅色紀錄
    {
        PORT_C |= _BV(pin - 8);
    }
}
```



1-2. 矩陣掃描




```
initMatrix_Red(22, 23, 24, 25, 26, 27, 28, 29, 37, 36, 35, 34, 33, 32, 31, 30);      //設定紅色
```

pinout 腳位↵

2-1. 貪食蛇物件

```
class snake                          //宣告貪食蛇類別
{
private:
    byte xy = 0, dir = 1, LinkedList_dir = 1; //上 1，搖桿現在狀態，上一次狀態，LinkedList 上一次狀態
    long setX = 0, setY = 0;                  //搖桿未移動準位
    byte analog_pin[3] = {0, 0, 0};           //貪食蛇類比控制腳(按鈕、X 值、Y 值)
    byte color = 1;                           //貪食蛇顏色
public:
    int speed = 300;                                                          //貪食蛇移動速度
    LinkedList<byte> snakeList = LinkedList<byte>();                          //https://github.com
/ivanseidel/LinkedList
    snake(byte a, byte b, byte c, byte d, byte e, byte f, byte g, byte h, byte i); //建立貪食蛇物件和設
定前三點的初始位置及搖桿控制腳位
    ~snake();                                                                 //遊戲結束時刪除貪食
蛇物件
    void Start_Matrix();                                                      //初始蛇的身體
    void joystick();                                                          //搖桿每次紀錄
    void linkedlist_add();                                                    //依照搖桿紀錄移動
    void snake_main(byte first);                                              //貪食蛇每次移動主程
式
    byte kill_snake(snake *all, byte first);                                  //判斷蛇是否死掉並回
傳數值
    void snake::lcdPrint_and_endset(byte first, byte kill_number, snake *Snake);    //遊戲結束時 LCD 提示
字元及設定結束時要做的事
    void apple(snake *all);                                                   //生成蘋果程式
    void cheak_hit_apple();                                                   //碰到蘋果程式
    void set_joystick();                                                      //初始設定搖桿準位
```

2-2. 貪食蛇物件

```
    void setcolor();                                                    //設定蛇的顏色
    void againcolor(byte last_color);                                   //設定蛇的顏色為上一
次遊玩時的顏色
    byte outputcolor();                                                 //輸出現在蛇顏色
    void reset();                                                       //設定蛇的初始方向及
遊戲開始設定
};
snake::snake(byte a, byte b, byte c, byte d, byte e, byte f, byte g, byte h, byte i) //建構子設定蛇的初
始位置及類比輸入腳位
{
    snakeList.add(a);  //貪食蛇第一個像素(X)
    snakeList.add(b);  //貪食蛇第一個像素(Y)
    snakeList.add(c);  //貪食蛇第二個像素(X)
    snakeList.add(d);  //貪食蛇第二個像素(Y)
    snakeList.add(e);  //貪食蛇第三個像素(X)
    snakeList.add(f);  //貪食蛇第三個像素(Y)
    analog_pin[0] = g; //搖桿按鈕腳
    analog_pin[1] = h; //搖桿類比輸入 X
    analog_pin[2] = i; //搖桿類比輸入 Y
}
```

3. 初始畫面

```
void StartInterface() //進行初始畫面
{
    clear_Matrix();                 //清除點矩陣
    Matrix_display_color = 7;       //設定矩陣要顯示的顏色顏色
    Matrix_display(Matrix_logo);    //顯示 logo
    lcd.setCursor(0, 0);            //設定游標為第一一列
    lcd.print("People number");     //lcd 印字
    lcd.setCursor(0, 1);            //設定游標為第二列
    lcd.print("up = 1 ,down = 2"); //lcd 印字
    play_people = 0;                //初始人數
    while (true)
    {
        if (digitalRead(A13) == 1 && digitalRead(A10) == 1) //將按紐放開
        {
            while (true)
            {
                if (analogRead(A12) > 700 || analogRead(A15) > 700) //搖桿往下
                {
                    if (play_people == 0)
                        lcd.clear();          //清除 LCD
                    lcd.setCursor(0, 0);      //設定游標
                    lcd.print("People = 2"); //顯示人數為 2
                    play_people = 2;          //設定人數為 2
                }
                if (analogRead(A12) < 300 || analogRead(A15) < 300) //搖桿往上
                {
                    if (play_people == 0)
                        lcd.clear();          //清除 LCD
                    lcd.setCursor(0, 0);      //設定游標
                    lcd.print("People = 1"); //顯示人數為 1
                    play_people = 1;          //設定人數為 1
                }
                if ((digitalRead(A13) == 0 || digitalRead(A10) == 0) && (play_people != 0))
                {
                    if (digitalRead(A13) == 0)       //如果左邊按鈕按下
                        LeftRightSwitch(1);          //設定遊玩按鈕為左邊(單人)
                    else if (digitalRead(A10) == 0) //如果右邊按鈕按下
                        LeftRightSwitch(0);          //設定遊玩按鈕為右邊(單人)
                    break;                           //結束
                }
            }
            break; //結束
        }
    }
}
```

4. 設定貪食蛇顏色

```
void snake::setcolor() //設定貪食蛇顏色
{
    clear_Matrix();                 //清除矩陣資料
    lcd.setCursor(0, 0);            //設定游標
    lcd.print("Select color");      //顯示 LCD
    lcd.setCursor(0, 1);            //設定游標
    lcd.print("Up = +  Down = -"); //顯示 LCD
    while (true)
    {
        if (digitalRead(analog_pin[0]) == 1) //先將按鈕放開
        {
            color = 1;      //設定預設顏色
            bool start = 0; //是否選擇為否
            while (true)
            {
                if (analogRead(analog_pin[2]) > 700) //搖桿往下
                {
                    start = 1;                              //是否選擇為是
                    lcd.clear();                            //清除 LCD
                    if (color > 1)                          //color 大於 1
                        color = color - 1;                  //color 減 1
                    Matrix_display_square(color);           //顯示預覽顏色
                    lcd.setCursor(0, 0);                    //設定游標
                    lcd.print("color:" + String(color, DEC)); //顯示 LCD
                    delay(500);                             //延遲 0.5 秒
                }
                if (analogRead(analog_pin[2]) < 300)
                {
                    start = 1;                              //是否選擇為是
                    lcd.clear();                            //清除 LCD
                    if (color < 7)                          //color 小於 7
                        color = color + 1;                  //color 加 1
                    Matrix_display_square(color);           //顯示預覽顏色
                    lcd.setCursor(0, 0);                    //設定游標
                    lcd.print("color:" + String(color, DEC)); //顯示 LCD
                    delay(500);                             //延遲 0.5 秒
                }
                if (digitalRead(analog_pin[0]) == 0 && start) //如果按下按鈕且是否選擇為是
                    break;                                  //結束
            }
            break;
        }
    }
}
```

5-1.Mega2560主程式

```
void loop()
{
    unsigned long move_J = 0, delay_time_snake = 0, delay_time_snake1 = 0; //搖桿、貪吃蛇 delay(millis()
    if (again == 0)                                                        //如果要重新開始
        StartInterface();                                                  //進行初始畫面
    if (play_people == 2)                                                  //遊玩人數為 2
    {
        snake Snake[2] = {snake(8, 8, 8, 9, 8, 10, 67, 68, 69), snake(4, 8, 4, 9, 4, 10, 64, 65, 66)};
//建立兩個 snake 的物件，https://crmne0707.pixnet.net/blog/post/317240238-c%2B%2B-%E9%A1%9E%E5%88%A5-
class-%E9%A1%9E%E5%88%A5%E7%89%A9%E4%BB%B6%E9%99%A3%E5%88%97
        access = Snake;
//設定物件指標
        if (again == 0)
//如果要重新開始
        {
            Snake->setcolor();       //重新選擇蛇的顏色
            (Snake + 1)->setcolor(); //重新選擇蛇的顏色
        }
        else //否則
        {
            Snake->againcolor(Last_snake0_color);       //設定貪吃蛇顏色為上一次遊玩的顏色
            (Snake + 1)->againcolor(Last_snake1_color); //設定貪吃蛇顏色為上一次遊玩的顏色
        }
        three_two_one();             //倒數 321
        Snake->apple(access);        //創造蘋果
        Snake->reset();              //貪吃蛇數據重置
        (Snake + 1)->reset();        //貪吃蛇數據重置
        Snake->Start_Matrix();       //初始蛇的身體
        (Snake + 1)->Start_Matrix(); //初始蛇的身體
        while (!end)                 //如果遊戲還沒結束
        {
            if (millis() - move_J > 1) //1ms 紀錄一次搖桿數據
            {
                move_J = millis();    //設定時間戳記
                Snake[0].joystick(); //紀錄右邊搖桿
                Snake[1].joystick(); //紀錄左邊搖桿
            }

            if (millis() - delay_time_snake > Snake[0].speed) //貪食蛇 1 移動
            {
                delay_time_snake = millis(); //設定時間戳記
                Snake[0].snake_main(0);      //右邊貪食蛇移動
            }

            if (millis() - delay_time_snake1 > Snake[1].speed) //貪食蛇 2 移動
            {
                delay_time_snake1 = millis(); //設定時間戳記
                Snake[1].snake_main(1);       //左邊貪食蛇移動
            }
        }
    }
```

5-2.Mega2560主程式

```
else if (play_people == 1) //遊玩人數為1
{
    snake Snake(8, 8, 8, 9, 8, 10, CTRL[0], CTRL[1], CTRL[2]); //建立一個 snake 物件
    access = &Snake;                                           //設定物件指標
    if (again == 0)                                            //如果要重新開始
        Snake.setcolor();                                      //重新選擇蛇的顏色
    else
        Snake.againcolor(Last_snake0_color);                   //設定貪吃蛇顏色為上一次遊玩的顏色
    three_two_one();                                           //倒數 321
    lcd.setCursor(0, 1);                                       //設定游標為第二列
    lcd.print("Record = " + String(Read_EEPROM(), DEC)); //顯示歷史最高分數
    Snake.apple(access);                                       //創造蘋果
    Snake.reset();                                             //貪吃蛇數據重置
    Snake.Start_Matrix();                                      //初始蛇的身體
    while (!end)                                               //如果遊戲還沒結束
    {
        if (millis() - move_J > 1) //1ms 紀錄一次搖桿數據
        {
            move_J = millis(); //設定時間戳記
            Snake.joystick();
        }
        if (millis() - delay_time_snake > Snake.speed) //貪食蛇移動
        {
            delay_time_snake = millis(); //設定時間戳記
            Snake.snake_main(0);
        }
    }
}
end_game(access); //遊戲結束
```

6.貪食蛇的主程式

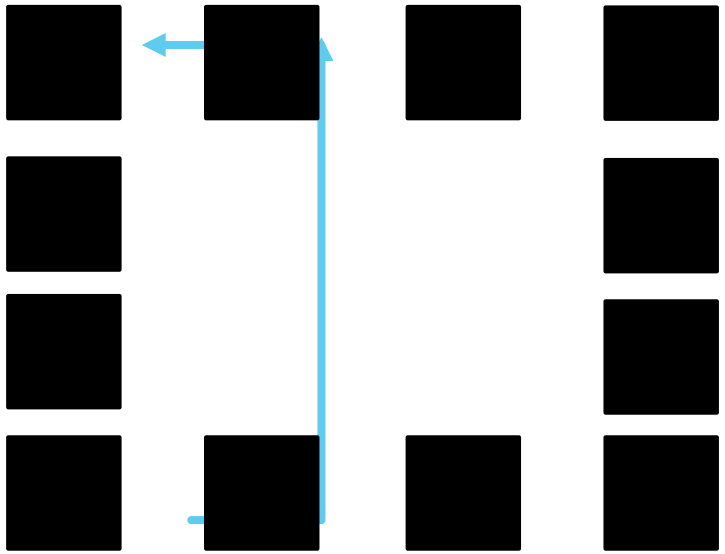
```
void snake::snake_main(byte first) //蛇每次動作的主程式
{
    linkedlist_add(); //蛇依照搖桿方向移動

    byte kill_number = kill_snake(access, first); //檢查遊戲是否結束
    if (kill_number == 5)                         //如果回傳值是 5(蛇沒有撞到任何東西)
    {
        Matrix[snakeList.get(1)][snakeList.get(0)] = color; //讀出來依照想要顯示的顏色顯示每個像素//
        cheak_hit_apple();                                  //是否碰到蘋果
    }
    else
        lcdPrint_and_endset(first, kill_number, access); //如果回傳值是其他的就停止遊戲
}
```

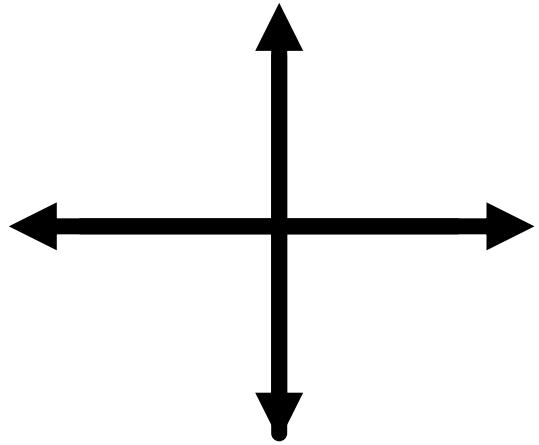
7-1.搖桿紀錄及貪食蛇動作

```
void snake::joystick() //搖桿紀錄
{
    if (digitalRead(analog_pin[0]) == 0) //按下按鈕時加快速度
        speed = 50;
    else
        speed = 300; //放開按鈕時速度變慢

    if (analogRead(analog_pin[2]) - setY < -300) //搖桿往上時
        dir = xy = 1;
    else if (analogRead(analog_pin[2]) - setY > 300) //搖桿往下時
        dir = xy = 3;
    else if (analogRead(analog_pin[1]) - setX < -300) //搖桿往右時
        dir = xy = 2;
    else if (analogRead(analog_pin[1]) - setX > 300) //搖桿往左時
        dir = xy = 4;
    else           //搖桿不動
        xy = dir; //設定方向為先前搖桿的移動方向
}
```

```
void snake::linkedlist_add()//依照搖桿紀錄的方向移動貪食蛇
{
    switch (xy)
    {
    case 1:                         //如果搖桿移動方向為上
        if (LinkedList_dir != 3)    //如果蛇的移動方向不是下
            LinkedList_dir = xy = 1; //向上移動
        else
            LinkedList_dir = xy = 3; //向下移動
        break;
    case 2:                         //如果搖桿移動方向為右
        if (LinkedList_dir != 4)    //如果蛇的移動方向不是左
            LinkedList_dir = xy = 2; //向右移動
        else
            LinkedList_dir = xy = 4; //向左移動
        break;
    case 3:                         //如果搖桿移動方向為下
        if (LinkedList_dir != 1)    //如果蛇的移動方向不是上
            LinkedList_dir = xy = 3; //向下移動
        else
            LinkedList_dir = xy = 1; //向上移動
        break;
    case 4:                         //如果搖桿移動方向為左
        if (LinkedList_dir != 2)    //如果蛇的移動方向不是又
            LinkedList_dir = xy = 4; //向左移動
        else
            LinkedList_dir = xy = 2; //向右移動
        break;
    default:
        break;
    }
```

7-2. 搖桿紀錄及貪食蛇動作





7-3.搖桿紀錄及貪食蛇動作

```
    switch (xy)
    {
    case 1:                                     //向上移動
        snakeList.add(0, snakeList.get(1) - 1); //加入 Y(值為前一元素的 Y 減一)
        snakeList.add(0, snakeList.get(1));     //加入 X(值為前一元素的 X)
        break;
    case 2:                                     //向右移動
        snakeList.add(0, snakeList.get(1));     //加入 Y(值為前一元素的 Y)
        snakeList.add(0, snakeList.get(1) + 1); //加入 X(值為前一元素的 X 加一)
        break;
    case 3:                                     //向下移動
        snakeList.add(0, snakeList.get(1) + 1); //加入 Y(值為前一元素的 Y 加一)
        snakeList.add(0, snakeList.get(1));     //加入 X(值為前一元素的 X)
        break;
    case 4:                                     //向左移動
        snakeList.add(0, snakeList.get(1));     //加入 Y(值為前一元素的 Y)
        snakeList.add(0, snakeList.get(1) - 1); //加入 X(值為前一元素的 X 減一)
        break;
    default:
        break;
    }
}
```

8-1.遊戲結束的判斷條件

```
byte snake::kill_snake(snake *all, byte first) //判斷是否結束遊戲
{
    for (int ii = 2; ii < snakeList.size(); ii += 2) //判斷是否撞到自己(將除了第一個像素點的像素都讀出來)
    {
        if (snakeList.get(0) == snakeList.get(ii)) //判斷 X 值是否一樣
        {
            if (snakeList.get(1) == snakeList.get(ii + 1)) //判斷 Y 值是否一樣
            {
                return 2; //如果都一樣就回傳 2
            }
        }
    }
    if (!(snakeList.get(0) >= 0 && snakeList.get(0) < 16 && snakeList.get(1) >= 0 && snakeList.get(1) <
16)) //如果出界
        return 0;
      //就回傳 0
    if (play_people == 2)
      //如果人數為 2
    {
        first = !first;
                    //將索引換成別人
```

8-2.遊戲結束的判斷條件

```
        if ((snakeList.get(0) == (all + first)-
>snakeList.get(0)) && (snakeList.get(1) == (all + first)->snakeList.get(1))) //如果兩個蛇的頭互相碰撞
        {
            return 3; //回傳 3
        }
        for (int ii = 2; ii < (all + first)->snakeList.size(); ii += 2) //將所有別人的像素點讀出來
        {
            if (snakeList.get(0) == (all + first)->snakeList.get(ii)) //如果 X 值一樣
            {
                if (snakeList.get(1) == (all + first)->snakeList.get(ii + 1)) //如果 Y 值一樣
                {
                    return 1; //回傳 1
                }
            }
        }
    }
    return 5;
}
```

9.依照顏色設定顯示想要顯示的內容

```
void Matrix_display(byte (*matrix)[16]) //將要顯示的內容送到矩陣顯示
{
    if (Matrix_display_color < 8) //如果顏色是符合要求的(8 設定為快速變換顯色)
    {
        for (byte ii = 0; ii < 16; ii++) //取出所有像素
        {
            for (byte jj = 0; jj < 16; jj++)
            {
                if (*(*(matrix + ii) + jj) != 0)                    //如果像素裡有資料
                    *(*(matrix + ii) + jj) = Matrix_display_color; //就改變顏色
                Matrix[ii][jj] = *(*(matrix + ii) + jj);           //並顯示出來
            }
        }
    }
}
```

10.創造蘋果

```
void snake::apple(snake *all) //創造蘋果
{
    bool repect;          //是否和貪食蛇的位置重複到
    if (play_people == 1) //如果遊玩人數是 1
    {
        score += 10;                                //進行加分
        lcd.setCursor(0, 0);                        //設定游標位置為(0,0)
        lcd.print("Score = " + String(score, DEC)); //LCD 印出分數
    }
    do
    {
        repect = 0;                                         //是否位置重複設定為 0
        apple_x = random(16);                               //隨機產生 0~15 的數當作 X 值
        apple_y = random(16);                               //隨機產生 0~15 的數當作 Y 值
        for (byte times = 0; times < play_people; times++) //查看玩遊戲的人數
        {
            for (int ii = 0; ii < (all + times)->snakeList.size(); ii += 2) //將貪食蛇的所有像素的 XY 值讀
出來
            {
                if ((all + times)->snakeList.get(ii) == apple_x && (all + times)-
>snakeList.get(ii + 1) == apple_y) //如果一樣
                {
                    repect = 1; //是否位置重複設定為 1
                    break;      //直接離開迴圈
                }
            }
        }
    } while (repect);             //如果重複就重複執行，沒重複就離開迴圈
    Matrix[apple_y][apple_x] = 1; //將符合資格的蘋果點亮
}
```

11-1.寫入資料到EEPROM或讀取資料

```
void Write_EEPROM_Score() //將分數寫入 EEPROM 裡
{
    String Score_string = "";                    //存放分數字串
    int divide = 999;                            //判斷位數
    for (byte i = 0; i < 4; i++, divide /= 10) //不足位進行補 0
    {
        if (score > divide) //判斷位數
        {
            for (byte j = 0; j < i; j++) //依照位數在數字前面補零
                Score_string += '0';
            Score_string += String(score, DEC); //最後將分數一起存入
            break;                               //離開迴圈
        }
    }
    for (byte i = 0; i < 4; i++) //讀取字串的 4 個字
    {
        EEPROM.write(i, Score_string[i]); //分別寫入字
    }
}
```

11-2.寫入資料到EEPROM或讀取資料

```
int Read_EEPROM()
{
    int history = 0;                //從 EEPROM 讀出來要儲存的地方
    int num = 1000;                 //讀出的值要乘上的權位
    for (byte i = 0; i < 4; i++) //讀出 4 個位元
    {
        history += (num * (EEPROM.read(i) - 48)); //將讀出的值做轉換再進行儲存
        num /= 10;                                //每做完一次權位除 10
    }
    return history; //回傳儲存的值
}
```

2-3外觀設計

2-3-1 Laserbox

我們外觀的材料是採用木材, 並且運用Laserbox這款軟體來設計模樣，進行雷射切割。

2-3-2 外部元件

金屬按鈕開關(push-buttonswitch)

我們是運用金屬按鈕開關來reset跟開關機。金屬按鈕開關是一種結構簡單，應用十分廣泛。在電氣自動控制電路中，用於手動發出控制信號以控制接觸器、繼電器、電磁起動器等。

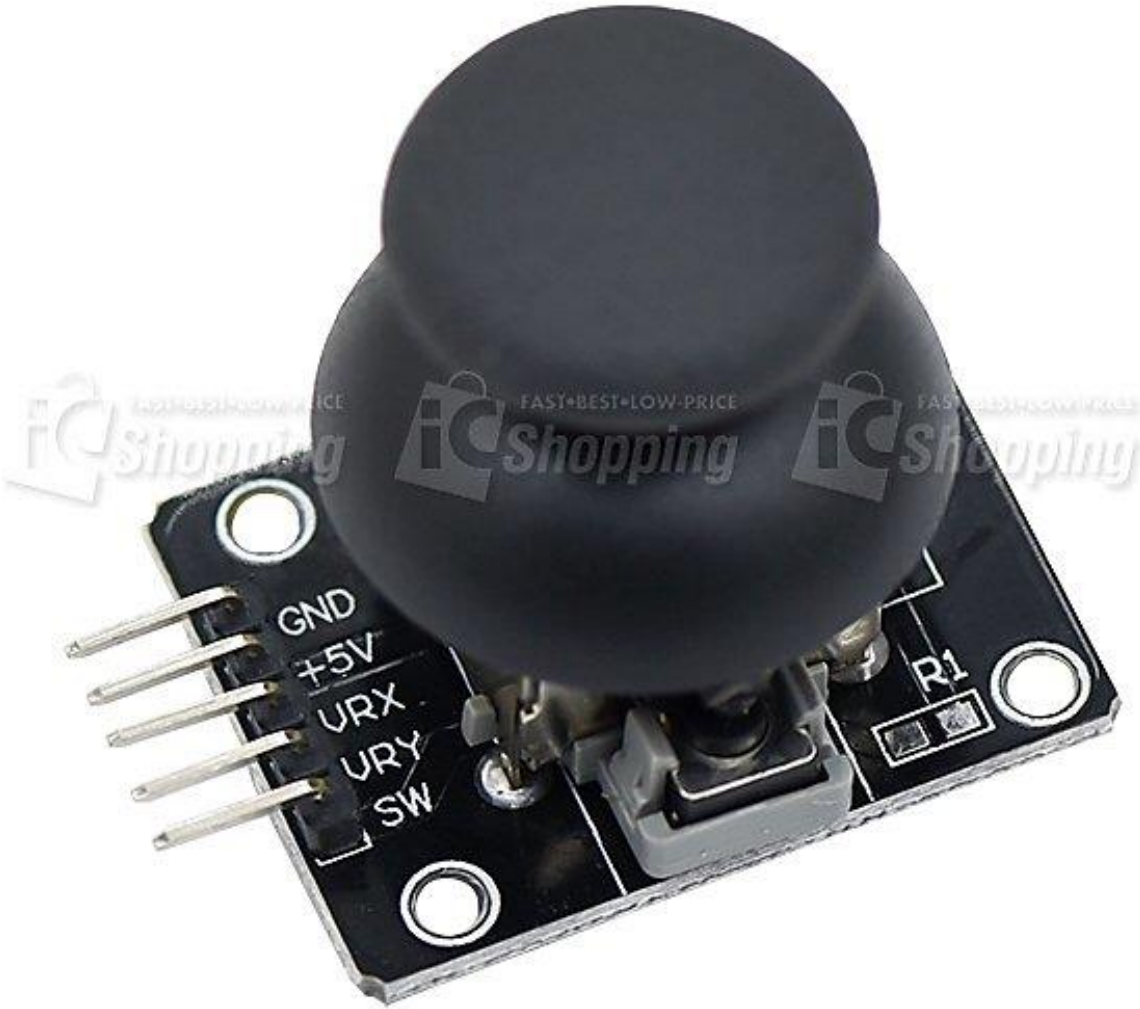
液晶顯示器(liquid-crystal display，LCD)

我們用液晶顯示器來顯示分數與功能列表

液晶顯示器為平面薄型的顯示裝置，由一定數量的彩色或黑白畫素組成，放置於光源或者反射環境光源。

類比搖桿

我們運用類比搖桿來控制貪吃蛇跟操控面板。類比搖桿，又稱模擬搖桿，是遊戲手把上用於控制方向的零件，通常用於控制遊戲中的人物角色移動方向，其作用等同於控制桿。 它突出於遊戲手把之上，通過與固定中心點的偏離位置確定輸入方向。



電源

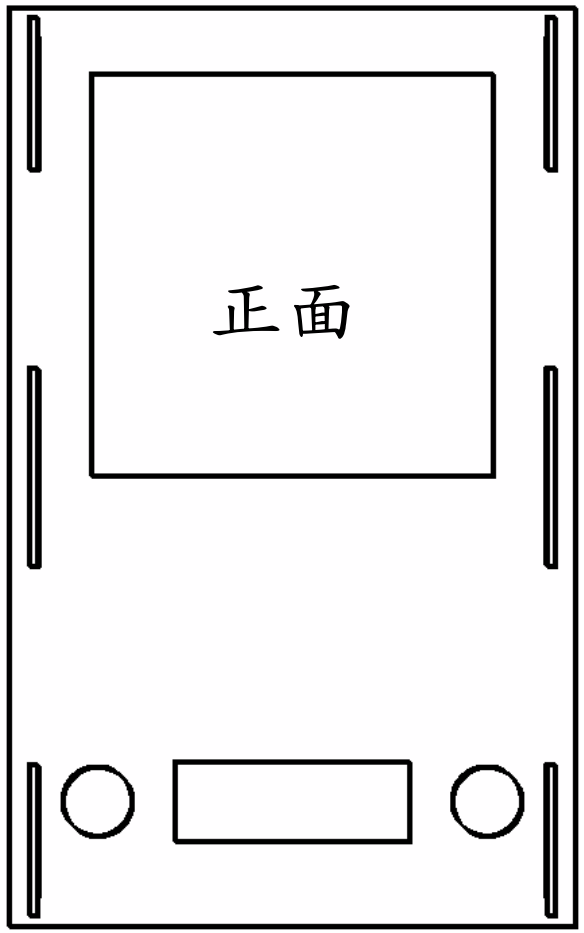
我們的電源供應方式是由18650鋰電池兩顆串聯組成。

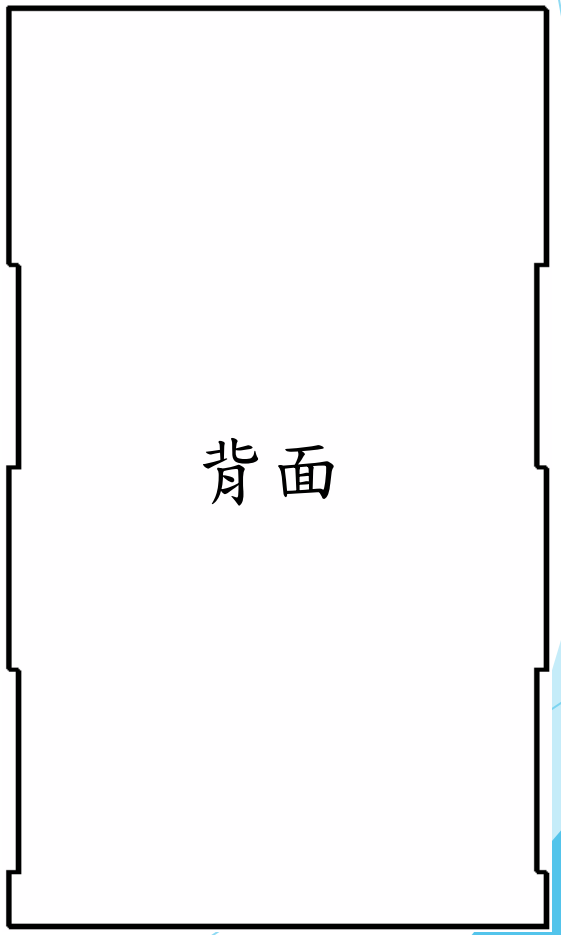
18650鋰電池，標準電壓都是3.6V或者3.7V，充滿電的時候是4.2V。

18650的意思是，直徑18毫米，長65毫米。

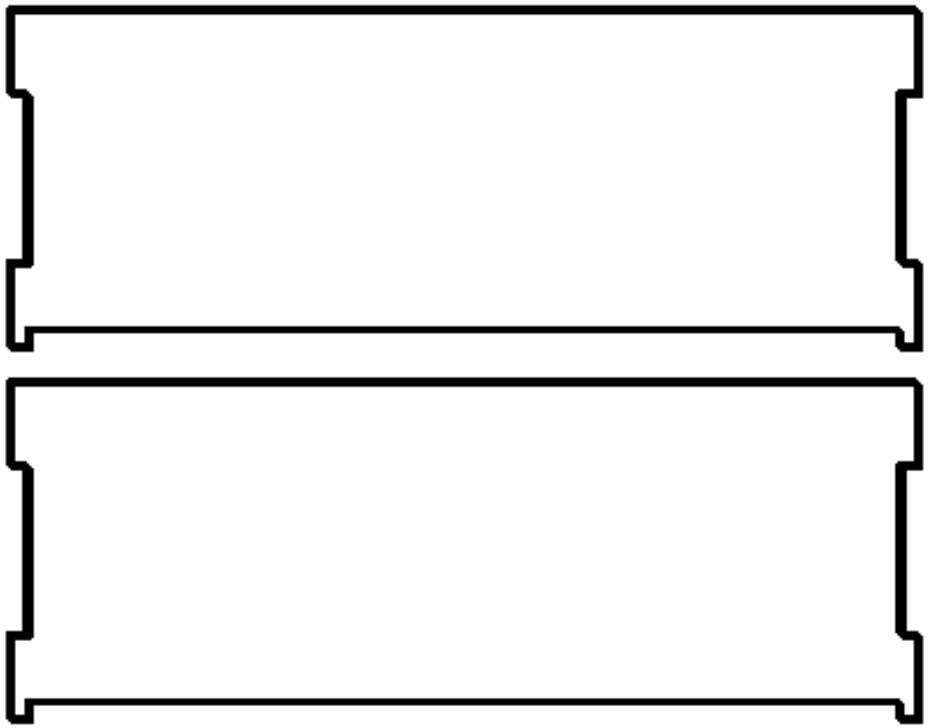
2-3-3 外觀

雷射切割設計圖

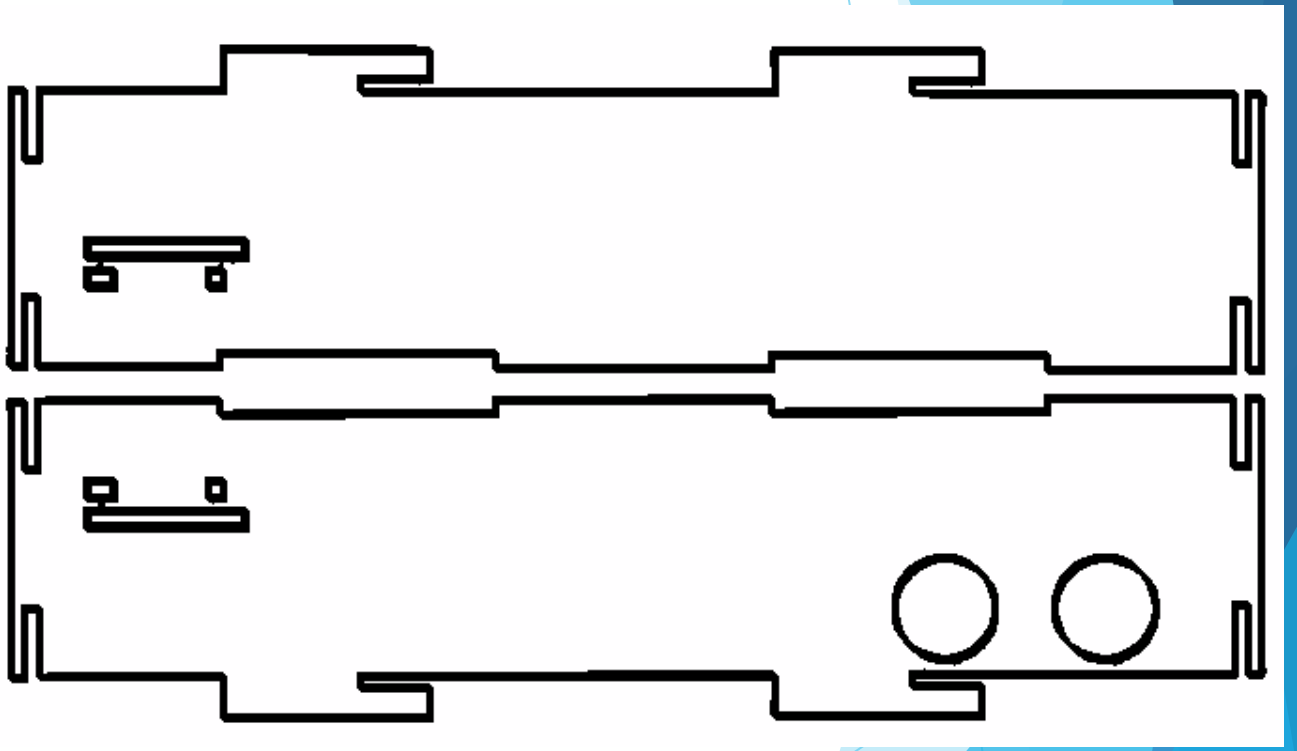




雷射切割設計圖



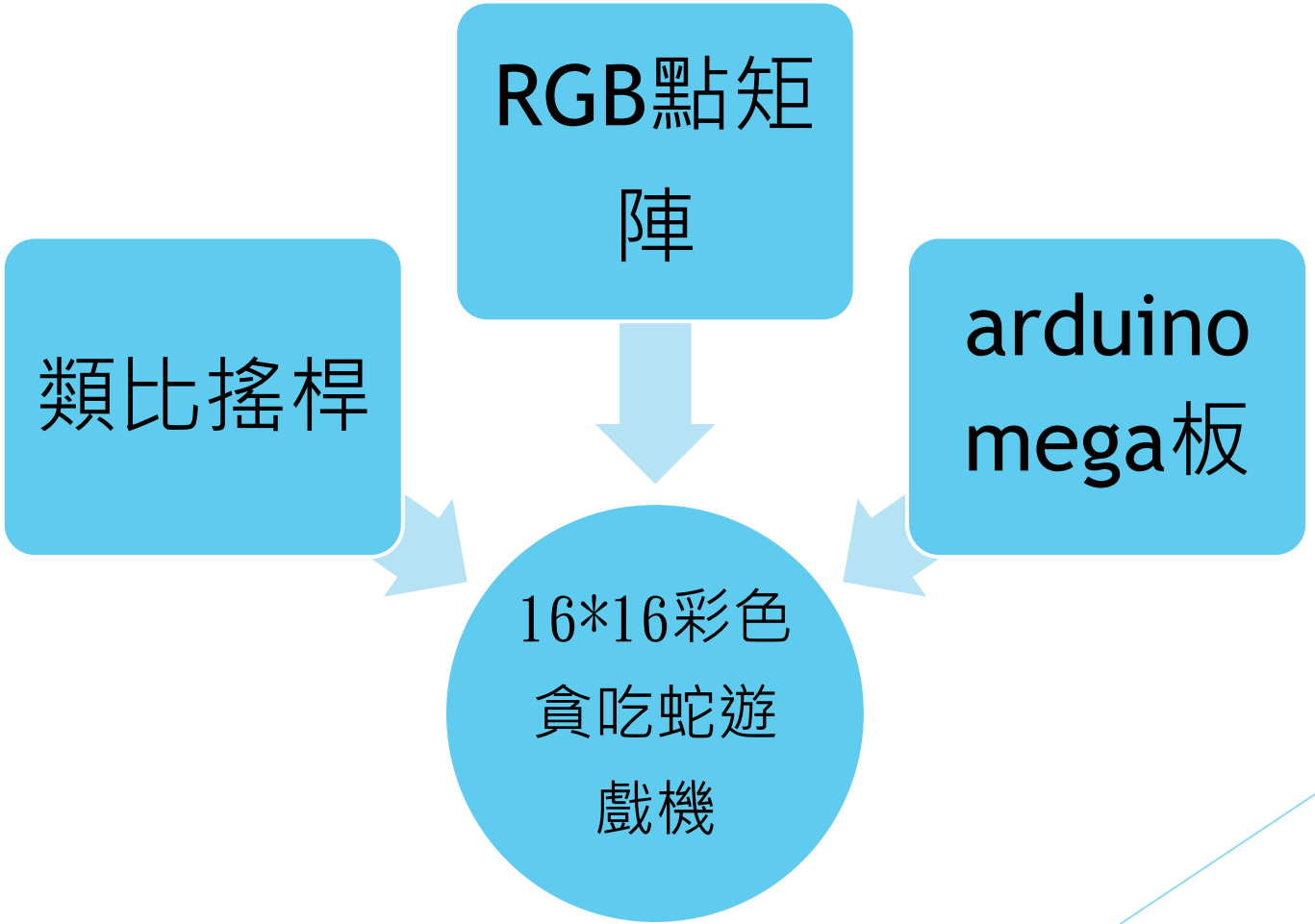
側面(前後)



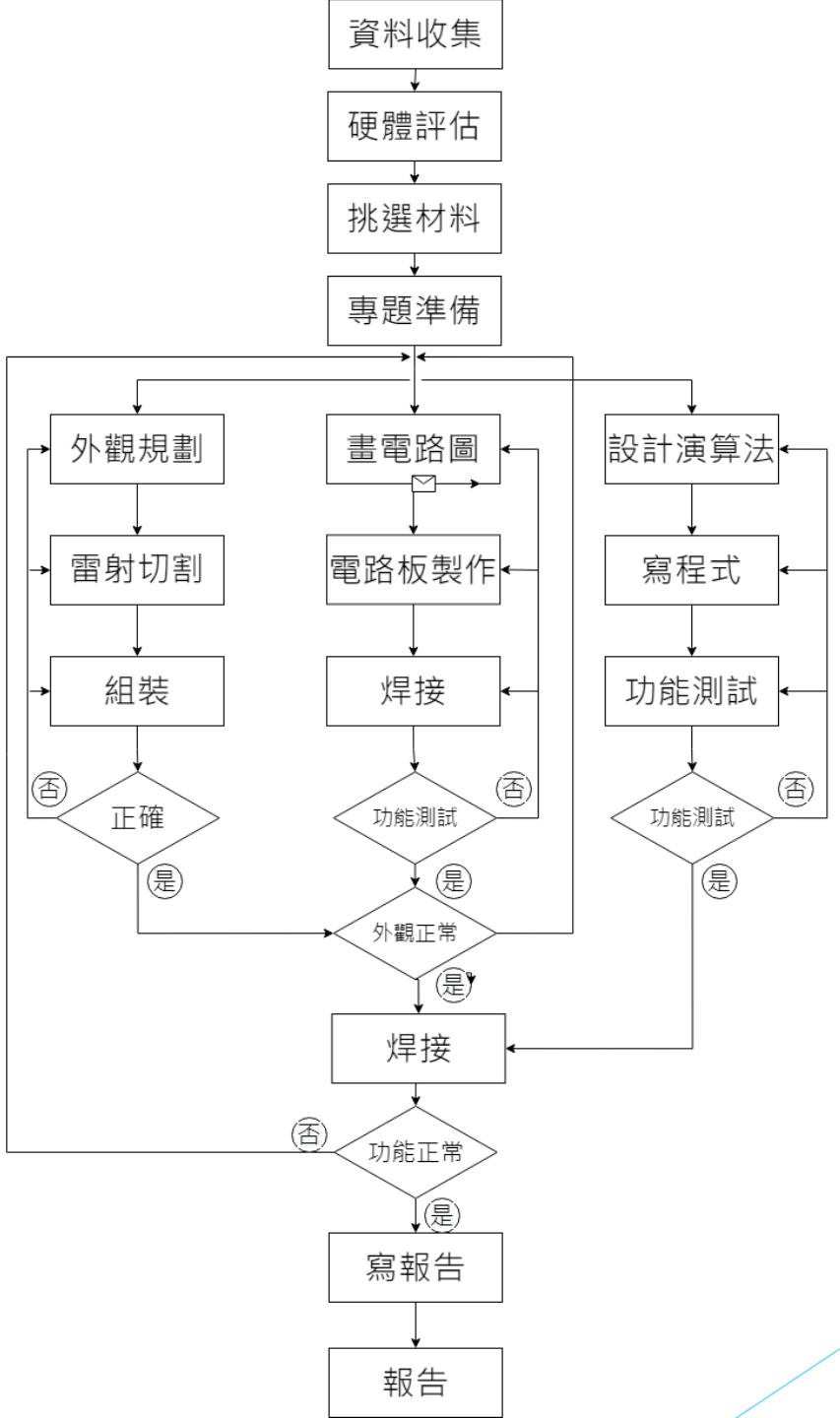
側面(左右)

第三章 專題準備

3-1系統架構



3-2流程圖


資料收集
硬體評估
挑選材料
專題準備
外觀規劃
畫電路圖
設計演算法
雷射切割
電路板製作
寫程式
組裝
焊接
功能測試
否
正確
功能測試
功能測試
否
否
是
是
是
外觀正常
是
焊接
否
功能正常
是
寫報告
報告

第4章 專題成果

4-1 問題與解決

遭遇的問題

- 1. 如果沒有SMD的元件，電路板會有很多鑽孔，不易佈線
- 2. Altium designer 內建的零件庫沒有我要用的原件
- 3. 零件庫中元件的腳位和實際的不同
- 4. 元件擺放的位置
- 5. 電阻太多太佔空間

遭遇的問題

- 6. 用來當電源的行動電源電壓高於電路設計的5V
- 7. 電路最大電流有可能會超過單顆穩壓IC的最大電流
- 8. 電路板顯像不大成功
- 9. 點矩陣亮度不夠
- 10. 掃描速度不夠
- 11. 點矩陣掃描有餘光

解決方式

- 1. MOSFET全採用SMD元件
- 2. 找內建零件庫中較相似的零件
- 3. 在零件屬性中手動調整腳位
- 4. 一個一個慢慢試，找到最好的排法
- 5. 把上拉和下拉電阻拆掉只留限流電阻，並把電阻全改成排阻
- 6. 電路多加穩壓IC
- 7. 把兩個穩壓IC並聯

解決方式

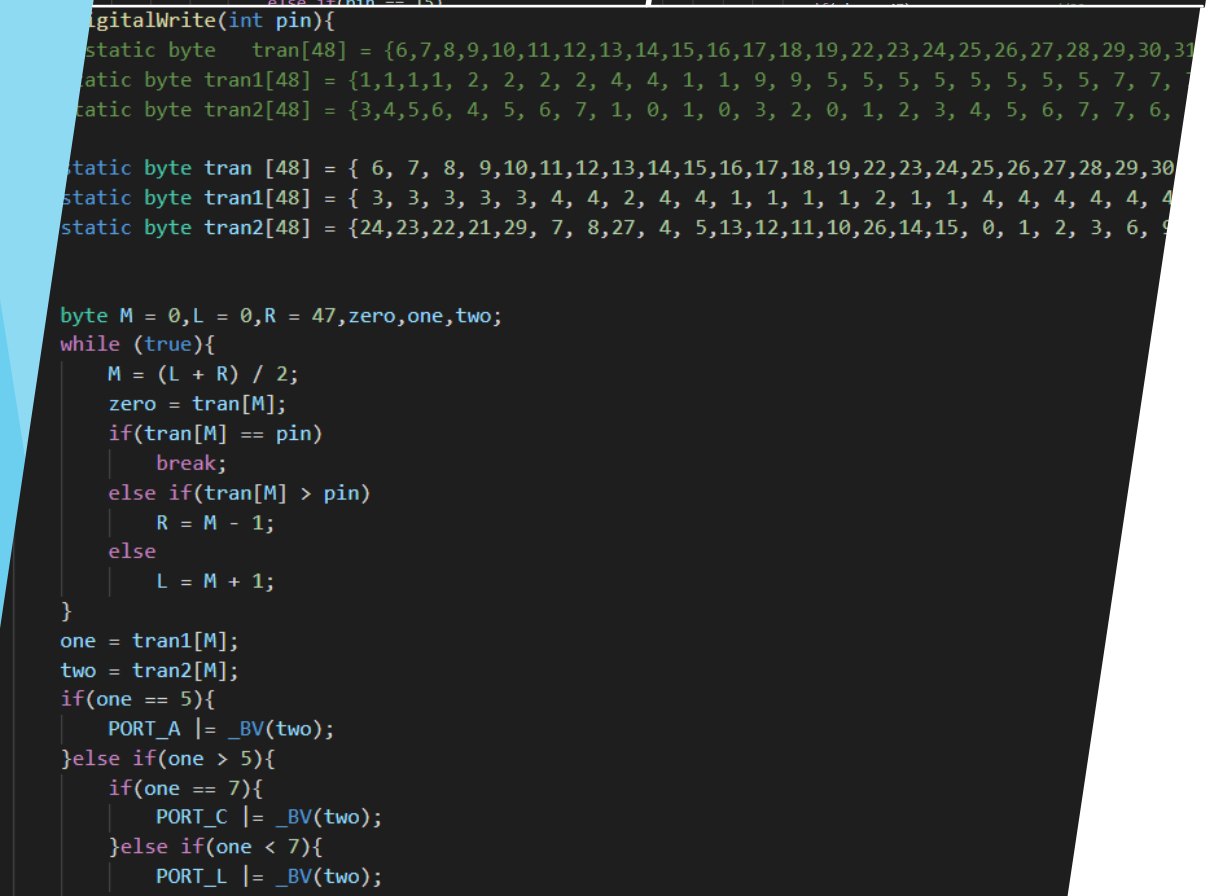
- 8.換另外一台正常的曝光機
- 9.更改程式寫法
- 10.更改程式寫法
- 11.目前無法解決


// Check if boxer is
if ($("#boxer").
return;
}
// Kill event
_killEvent(e);
// Cache internal
data = $.extend(
$window:
$body:
$target:
$object:
visible: false,
resizeTimer: null,
touchTimer: null,
gallery: {
active: false
軟體問題

```
igitalWrite(int pin){
(pin < 30){
  if(pin < 17){
     if(pin < 14){
        if(pin >= 6 && pin <= 9){
           for(int c = 3;c < 7;c++){
              if((c + 3) == pin){
                 PORT_H |= _BV(c);
                 break;
              }
           }
        }else if(pin >= 10 && pin <= 13){
           for(int c = 4;c < 8;c++){
              if((c + 6) == pin){
                 PORT_B |= _BV(c);
                 break;
              }
           }
        }
     }else{
        if(pin == 14)
           PORT_J |= 0x02;
        else if(pin == 15)
```



```
talWrite(byte pin){
n == 31)                          //23
ORT_C |= _BV(6);
if(pin > 31){
if(pin == 43)                     //35
   PORT_L |= _BV(6);
else if(pin > 43){
   if(pin == 49)                  //41
      PORT_L |= _BV(0);
   else if(pin > 49){
      if(pin == 52)               //44
         PORT_B |= _BV(1);
      else if(pin > 52){
         if(pin == 59)
            PORT_F |= _BV(5);     //46
         else if(pin > 59)
            PORT_F |= _BV(6);
         else
            PORT_B |= _BV(0);
      }else{
         if(pin == 50)            //42
            PORT_B |= _BV(3);
         else
            PORT_B |= _BV(2);
      }
   }else{
      if(pin == 46)               //38
         PORT_L |= _BV(3);
      else if(pin > 46){
```



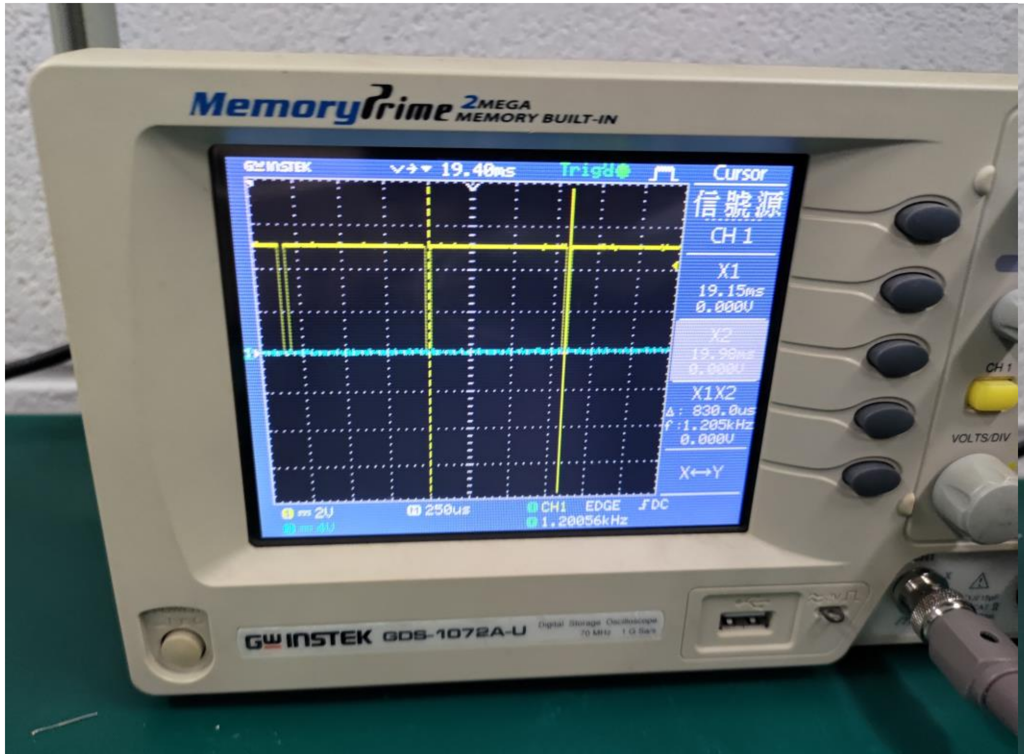
```
igitalWrite(int pin){
static byte   tran[48] = {6,7,8,9,10,11,12,13,14,15,16,17,18,19,22,23,24,25,26,27,28,29,30,31
atic byte tran1[48] = {1,1,1,1, 2, 2, 2, 2, 4, 4, 1, 1, 9, 9, 5, 5, 5, 5, 5, 5, 5, 5, 7, 7,
tic byte tran2[48] = {3,4,5,6, 4, 5, 6, 7, 1, 0, 1, 0, 3, 2, 0, 1, 2, 3, 4, 5, 6, 7, 7, 6,

tatic byte tran [48] = { 6, 7, 8, 9,10,11,12,13,14,15,16,17,18,19,22,23,24,25,26,27,28,29,30
static byte tran1[48] = { 3, 3, 3, 3, 3, 4, 4, 2, 4, 4, 1, 1, 1, 1, 2, 1, 1, 4, 4, 4, 4, 4,
static byte tran2[48] = {24,23,22,21,29, 7, 8,27, 4, 5,13,12,11,10,26,14,15, 0, 1, 2, 3, 6,


byte M = 0,L = 0,R = 47,zero,one,two;
while (true){
    M = (L + R) / 2;
    zero = tran[M];
    if(tran[M] == pin)
        break;
    else if(tran[M] > pin)
        R = M - 1;
    else
        L = M + 1;
}
one = tran1[M];
two = tran2[M];
if(one == 5){
    PORT_A |= _BV(two);
}else if(one > 5){
    if(one == 7){
        PORT_C |= _BV(two);
    }else if(one < 7){
        PORT_L |= _BV(two);
```

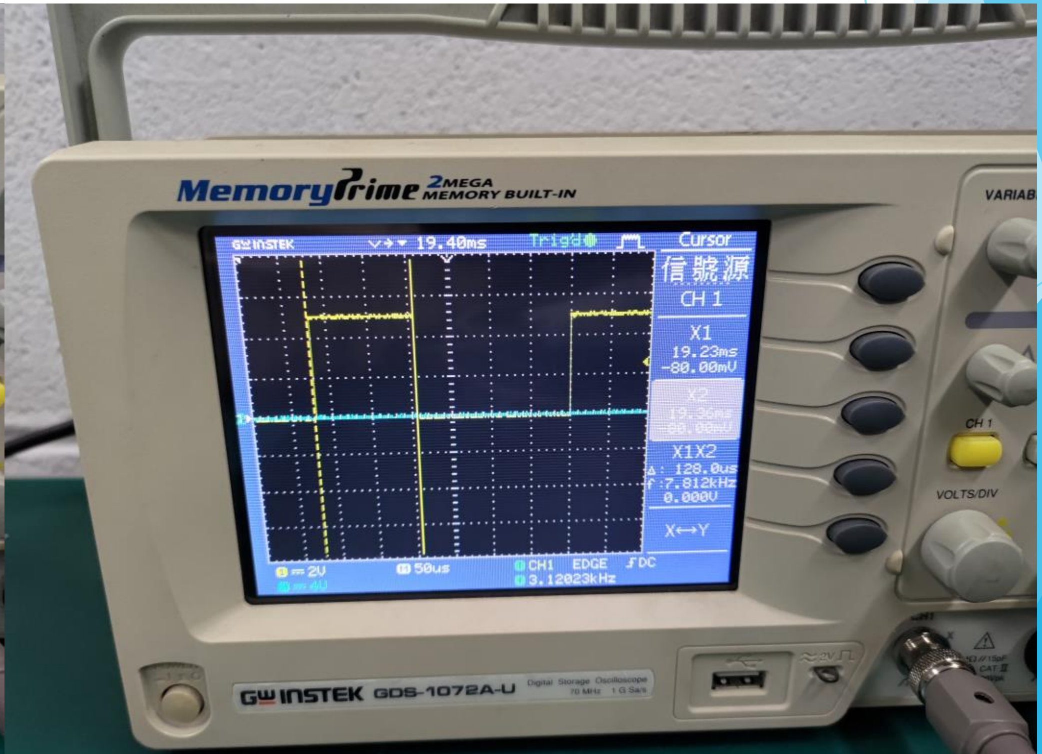
1-1.掃描速度及亮度

1-2. 掃描速度及亮度

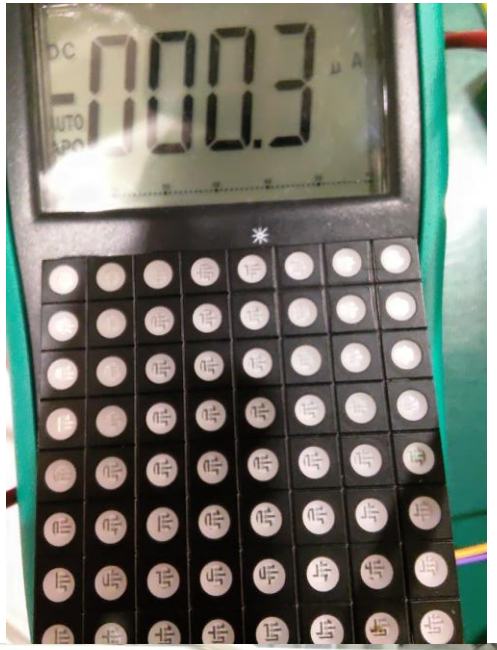
```
byte Matrix[16][16] = { //顯示在矩陣上的內容
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
```

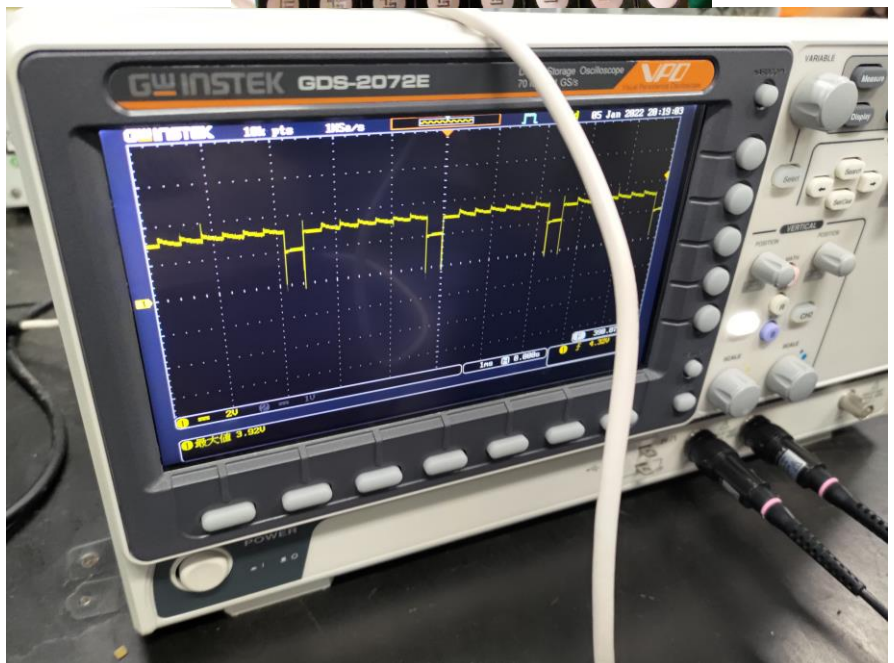
```
clear_led(); //掃描及資料線腳位暫存器紀錄歸零

byte S_LED = 0; //暫存像素資料

for (byte ii_data = 0; ii_data < 16; ii_data++) //掃描16顆LED
{
```

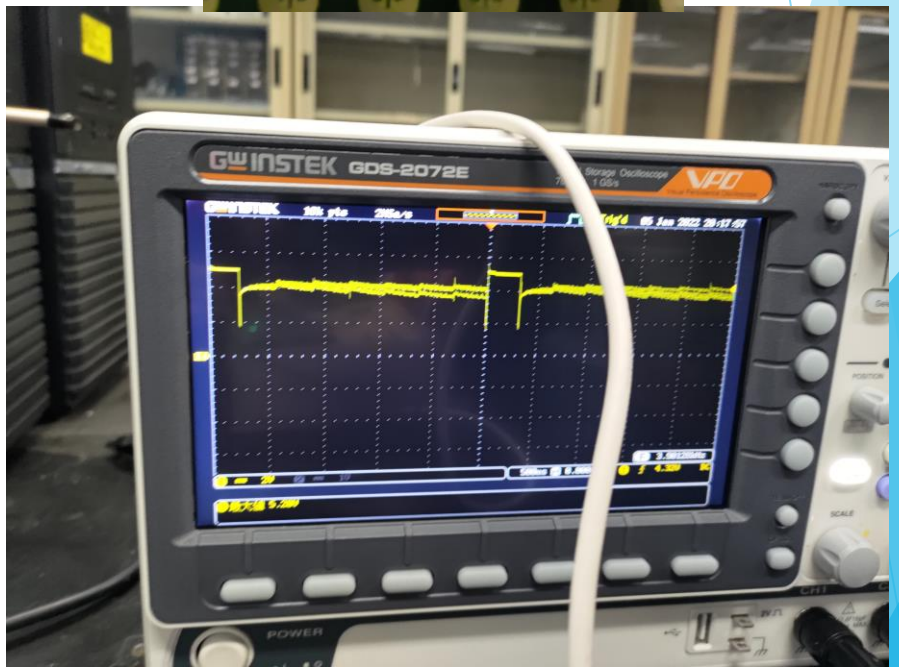




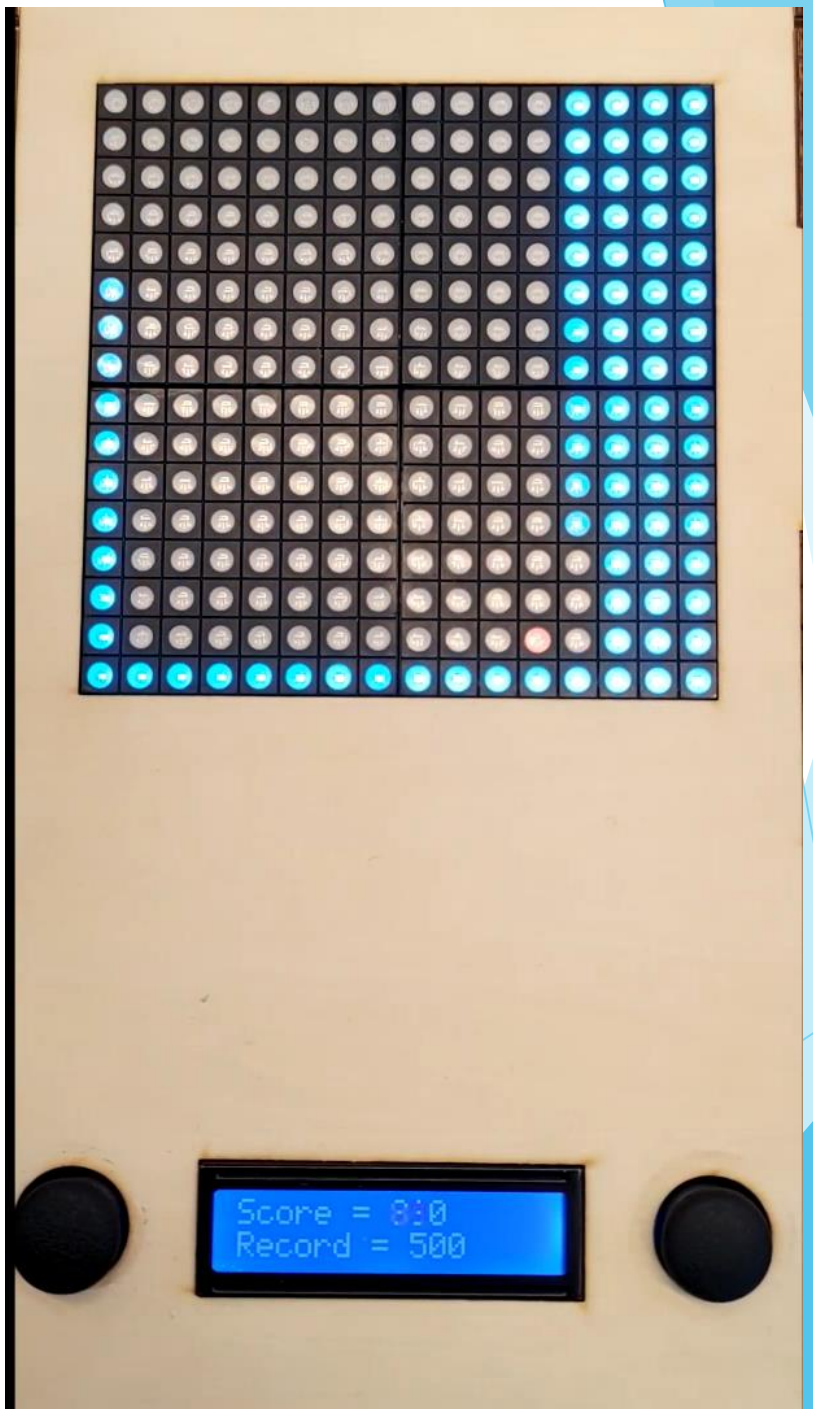
2. 點矩陣掃描有餘光


DC
-000.3
μA
AUTO


GW INSTEK GDS-2072E
Storage Oscilloscope
VPO
VARIABLE
VERTICAL
POWER

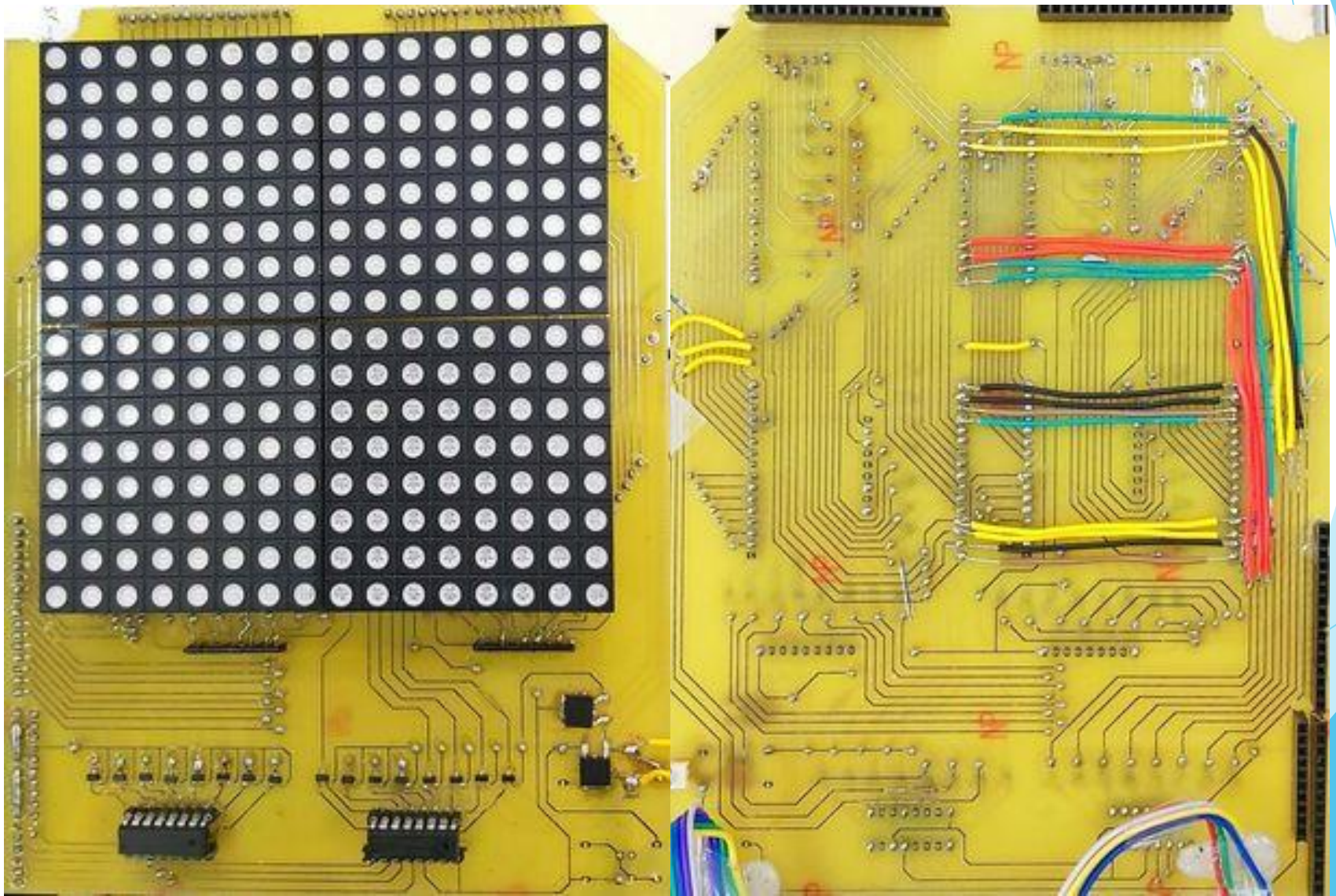

GW INSTEK GDS-2072E
Storage Oscilloscope
VPO
POWER

3. 當貪食蛇吃滿時



4-2 成果

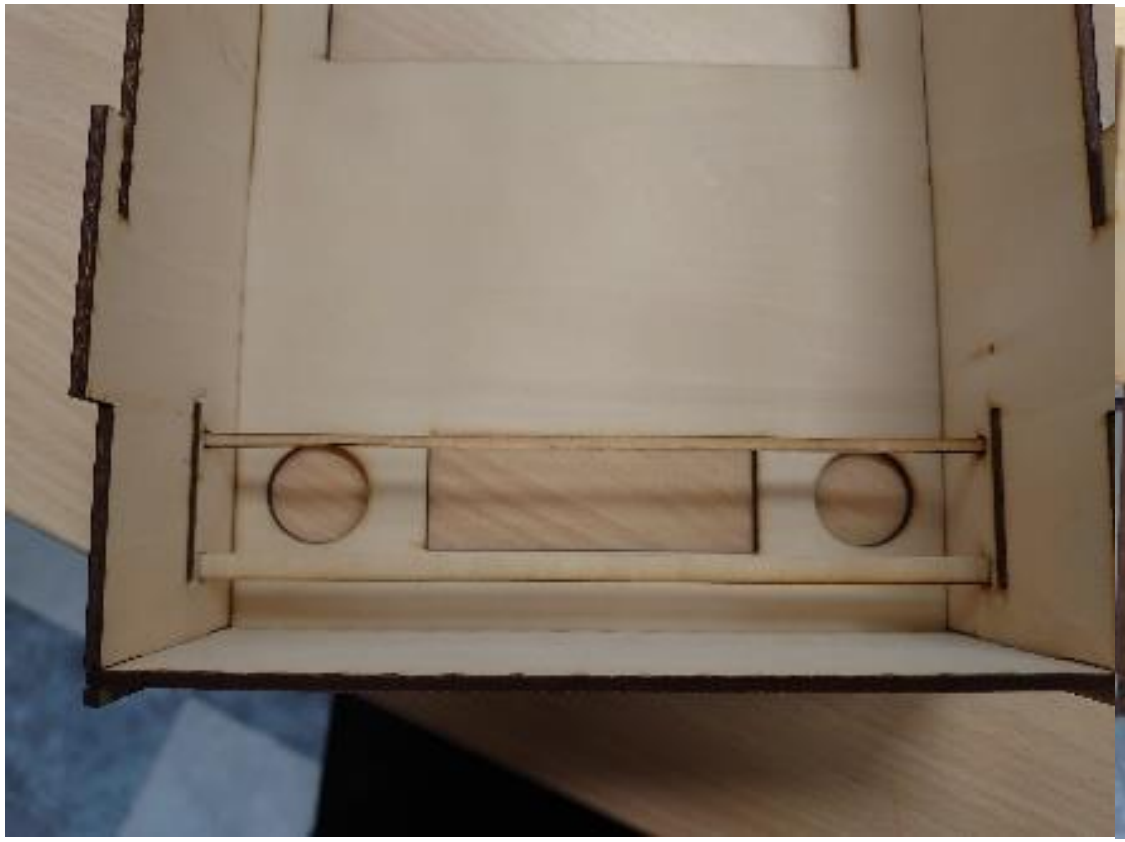
電路板成品

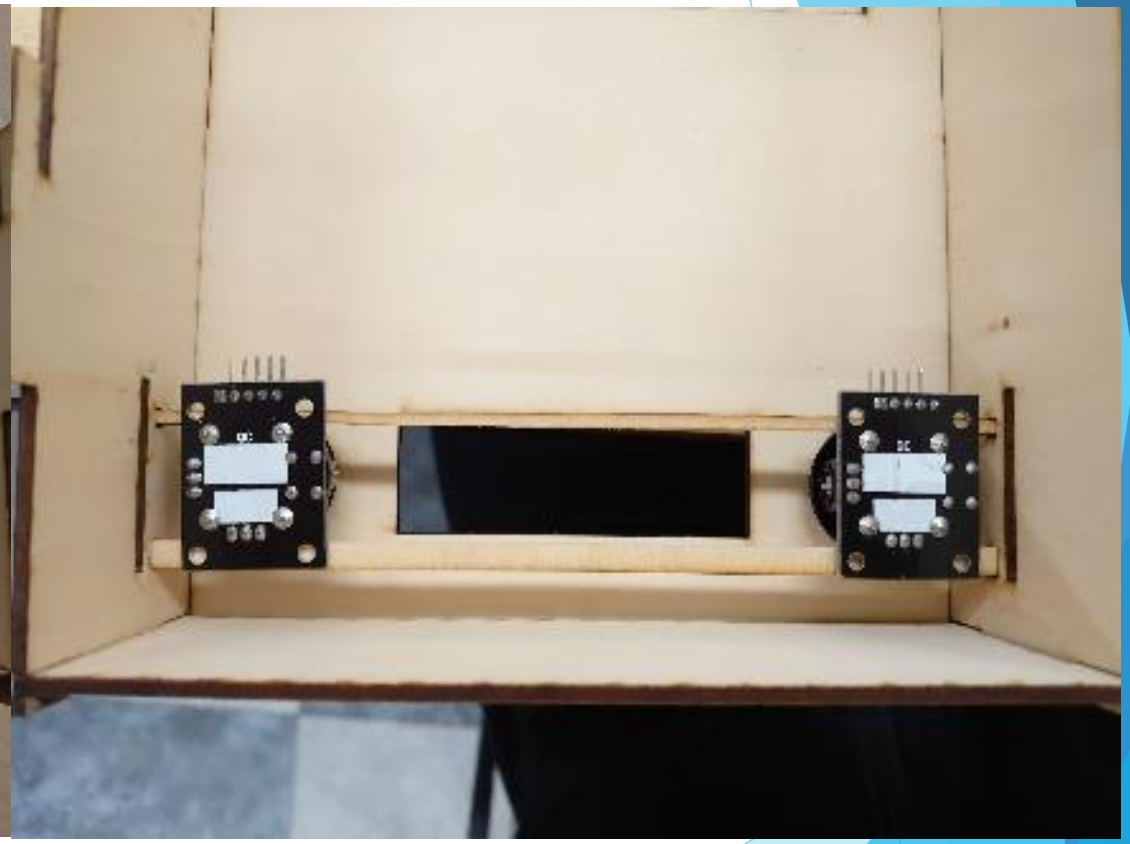


實體外觀

實體外觀

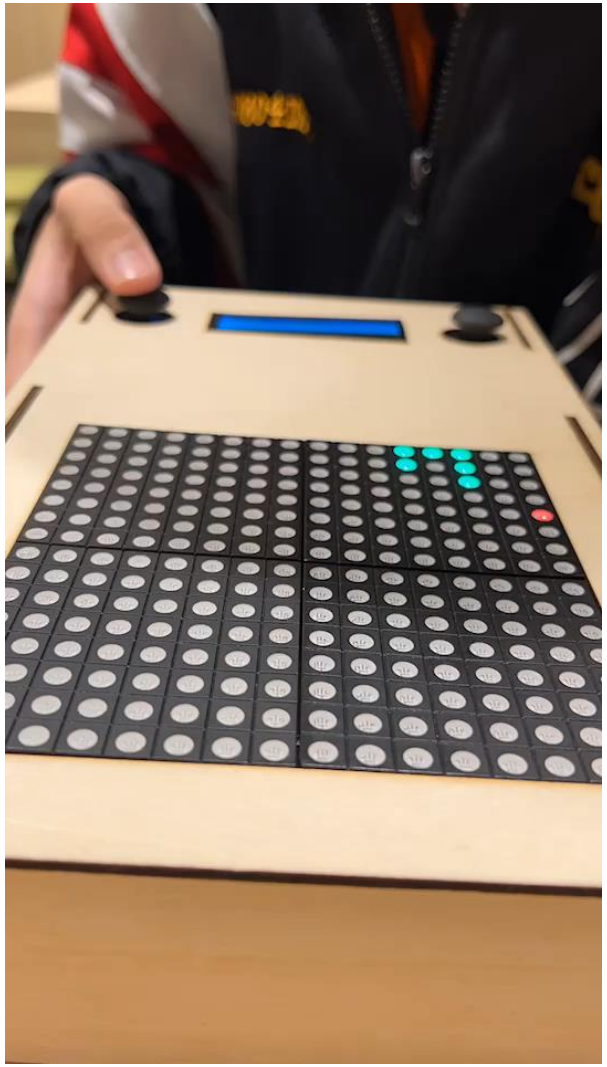
實體外觀

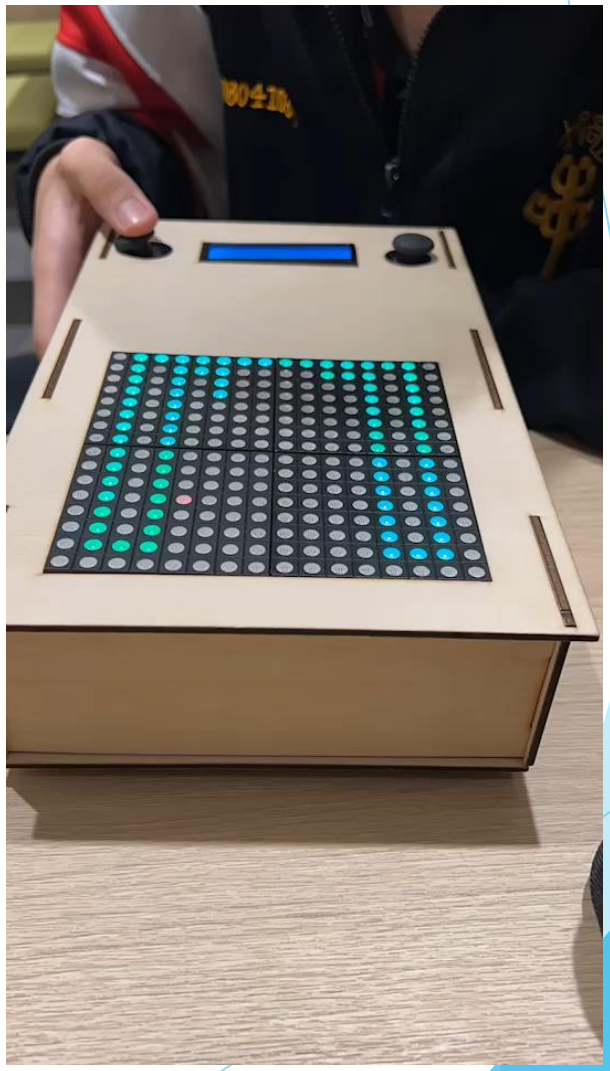




成品外觀

成品影片(單人)





第五章結論與建議

5-1結論與建議

經過這個專題實習課程，我們要給想做遊戲機當專題的學弟妹一些建議，在設計電路圖時，能用SMD儘量用SMD，比較好焊也比較方便。在洗電路板時曝光機的使用要很小心，不然電路板會容易洗壞。外觀的製作最好使用堅固的材料，因為我們是使用木材，而且是用卡榫來固定，所以在交接處常常斷掉，就又再用一個新的。最後建議大家好好分工合作，不要為了小事起爭執，容易讓進度落後。

QA

謝謝大家

THANKS!