國立台南大學附屬高級中學

學術學程 自然科

專題研究報告

Department of Sciences

Branch of Academic

The Affiliated Senior High School of

National University of Tainan

Thematic Study Report

互動式五子棋遊戲於 Arduino 上之實作

Interactive gobang game on Arduino Uno board

隊名：Team jian xia yu de sheng yin

組長：丁文瑩

組員：謝柏毅 王柏崴 李苡瑄 吳昱霖

指導教授： 蕭學宏　教授、張銘傑　老師

中華民國一一〇年六月

摘要

“當我設計出一套程式，要如何跟現實世界作結合”，這是時常令我疑惑的問題。當我有一套很有技術的程式，我要怎麼做能讓它不再只是在電腦中運行的「軟體」，而是看的到、摸的到且可以簡化、處理我們生活中從瑣事到需要精密操作的「硬體」。

就像達文西手臂（達文西外科手術系統，da Vinci Surgical System）一樣，當程式與機械結合時，能讓原本就是為了省力而發明出的機械之能力有所成長，可以進行更細膩的操作，錯誤率也會降低。而人工智慧（Artificial Intelligence）相當於一顆人造大腦，能自發性學習及進行快速演算，並在短時間內處理大量的物體情報，當人工智慧與機械手臂等科學技術產物結合時，定能讓其之能力有更大幅度的成長，且能夠更加簡化原本繁複的流程。

Arduino 專案始於 2003 年，為一義大利伊夫雷亞互動設計研究所（Interaction Design Institute Ivrea）學生之專案，其目的是為了讓初學者及專業人員可以用較低成本的方法，以建立使用感測器等電子零件與環境相互作用的裝置執行器[1]。此微控制器的發明，讓一些程式愛好者能訓練程式與電路板結合的能力，且不需要高額的經費，各種硬體上的其他限制例如：「隨機存取記憶體（Random Access Memory，RAM）有時會因不足導致目標呈現延遲或失效」等，大部分都能利用擴充的方式解決，同一塊電路板還可重複使用，這些特點讓一些有想法的設計者能立即實驗，且負擔較小。

本研究將使用 Arduino 公司的 Arduino Uno 製作簡易互動式的五子棋遊戲，並藉由此次實作，除了增進焊接、電路安排等電路工程能力，同時也會精進程式碼中，構建自訂函式的能力及使用遞迴的邏輯，讓不論是「硬體」或「軟體」方面的能力都有所成長。

組員分工及聲明

壹、成品目標發想

（一）負責人：丁文瑩

貳、程式設計

（一）負責人：丁文瑩

（二）本研究之程式設計構想、程式碼皆由 丁文瑩 自行繪成、編程編寫，且無從任何管道擷取或轉載部分或全部之程式碼或架構。

（三）本研究程式部分所使用之函式庫相關不受限於上述。

參、電路設計

（一）負責人：謝柏毅、李苡瑄

（二）由 小組成員 共同討論、設計。

肆、美工設計

（一）負責人：吳昱霖、王柏崴

（二）由 小組成員 共同討論、設計。

伍、購買材料

（一）負責人：丁文瑩

（二）所用材料除線材、Arduino Uno 電路板、電阻及傳輸線外，其他皆於於蝦皮購物網站上購入。

陸、資料統整

（一）負責人：丁文瑩

（二）圖 2-4 及圖 3-5 至 3-7 因輸出時解析度大，閱讀時可能需要一段時間才可轉變至清晰。

（三）資料統整包括報告編寫、設計及 pdf 檔案發佈皆為此項目負責人自行完成。

目錄

摘要 I

組員分工及聲明 II

目錄 III

圖目錄 V

第一章　緒論

第一節 研究動機 1

第二節 研究策略 1

第三節 研究所需材料及準備 2

第四節 研究範圍及限制 3

第二章 構想與構圖

第一節 成品部分 4

第二節 電路部分 5

第三節 程式部分 7

第三章 實作與測試

第一節 電路部分 8

第二節 程式部分 10

第四章 展示 16

第五章 研究心得 17

第六章 參考文獻..... 18

第七章 附錄 19

圖目錄

圖 1-1［排針與模組間的焊接］ 2

圖 2-1［成品構想與構圖］ 3

圖 2-2［電路構想與構圖］ 4

圖 2-3［WS2812-B 電路參考圖］ 6

圖 2-4［程式構想］ 7

圖 3-1［模組電路實作（一）］ 8

圖 3-2［模組電路實作（二）］ 8

圖 3-3［模組電路測試成功］ 8

圖 3-4［上拉電阻按鈕］ 9

圖 3-5［About: Check］ 11

圖 3-6［About: Call］ 12

圖 3-7［About: loop］ 14

圖 5-1［程式版本差異舉例］ 17

第一章　緒論

第一節 研究動機

在現今社會，人工智慧已成為許多程式設計師、科技愛好者，甚至是科學家們不斷研究、突破的領域。其中運用到許多在學習程式設計時，時常忽略或沒有弄清楚的程式概念，例如本研究的核心：遞迴 (Recursion) 。

遞迴在電腦科學領域中的本質即為縮小問題規模，把規模大的、較難解決的問題變為較簡單、容易解決，並且在小到一定程度時，直接得到其解，從而得到原問題之解。

遞迴演算法因其運作方式大多為函式 (Function) 間互相或自身的重複呼叫 (Call) ，於編程時使用該演算法能一定程度的精簡程式碼，更可透過各函式所代表的涵義以增加程式碼整體之可讀性。值得注意的是，C 編譯器處理函式呼叫時是使用堆疊 (Stack) 來保存資料的，當主呼叫函式呼叫一個被呼叫函式時，此編譯器會將堆疊中的實質參數 (Argument，譯引數) 彈出 (Pop) ，並指定值給形式參數 (Parameter，譯參數) ，同時利用此一堆疊保存被呼叫函式之區域變數 (Local variables)，將兩者及堆疊頂的指標 (Pointer) 壓入 (Push) 後開始執行被呼叫函式，於執行結束時再彈出。從上述明顯可以得知一個遞迴的缺點，遞迴的內部實現需要消耗額外的空間與時間，如果呼叫的層次太深，可能導致堆疊溢位 (Stack overflow)，進而讓程式執行崩潰[2]。

本研究原先是因想精進在做程式之編程時，函式遞迴及資料儲存並調用的能力而設計的，剛好有這個機會可以讓設計出來的程式在生活中被實際運用，又因最近時常與朋友下五子棋，於是決定讓本實驗藉由實作互動式五子棋遊戲，以達成提升遞迴邏輯之目標。

第二節 研究策略

當程式要與電路板結合時，相信電路板上的電路安排才是整體的基礎，如果程式設計師沒有完全了解此電路板及裝置間的電路安排，在進行編程時就很容易受到諸多限制，也會碰到許多窒礙難行的地方，所以本研究將以先設計電路後設計程式為主。

第三節 研究所需材料及準備

壹、設備及材料

（一）電路部分

1. Arduino Uno * 1
2. WS2812-B-(8*1) * 8
3. USB Cable Type A/B * 1
4. 杜邦線 * 48
5. 麵包板 * 4
6. 按鈕 * 5
7. 230 ± 5% Ω 電阻 * 5
8. 焊料乙條
9. 烙鐵乙支
10. 排針 * 32

（二）程式設計部分

1. 筆記型電腦 (OS: Ubuntu 20.04 LTS)
2. Visual Studio Code (VSCode)
3. Visual Studio Code extension for Arduino (By Microsoft)
4. FastLED Library (From GitHub) [3]
5. Git

貳、準備

（一）將排針與 WS2812-B 燈條焊接，如附圖。

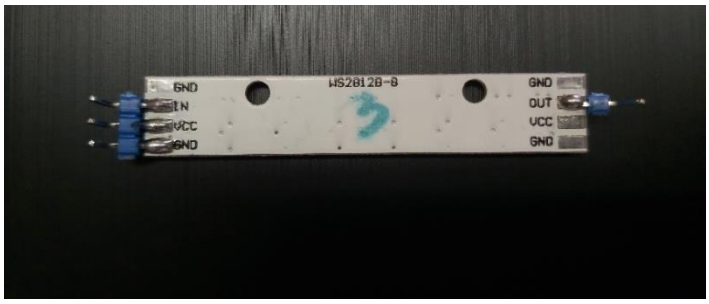

圖 1-1 排針與模組間的焊接

（二）將 Arduino Uno 與電腦連接埠連接，確定電腦端有偵測到。

（三）安裝 VSCode、Arduino extension

（四）下載 FastLED library 並確認 FastLED 標頭檔可以被成功含括

第四節 研究範圍及限制

本研究於電路構建中有使用上拉電阻 (Pull-up resistors) 的構造，因其特性及運作原理，當電流通過時，會額外耗損能量，且可能會引起其輸出之邏輯電平的延遲。上述並不在本研究預想的範圍內，故暫不探討其引發的問題，於程式碼中也沒有加入預防機械彈跳 (Bounce) 的機制。

因考慮所有組員之經濟狀況，本研究規模以實驗可以成功為原則，尚未把棋盤規模擴大至實際大小，後續使用者若上述條件許可，請自行更動程式碼中之相關常數。

第二章　構想與構圖

第一節　成品部分

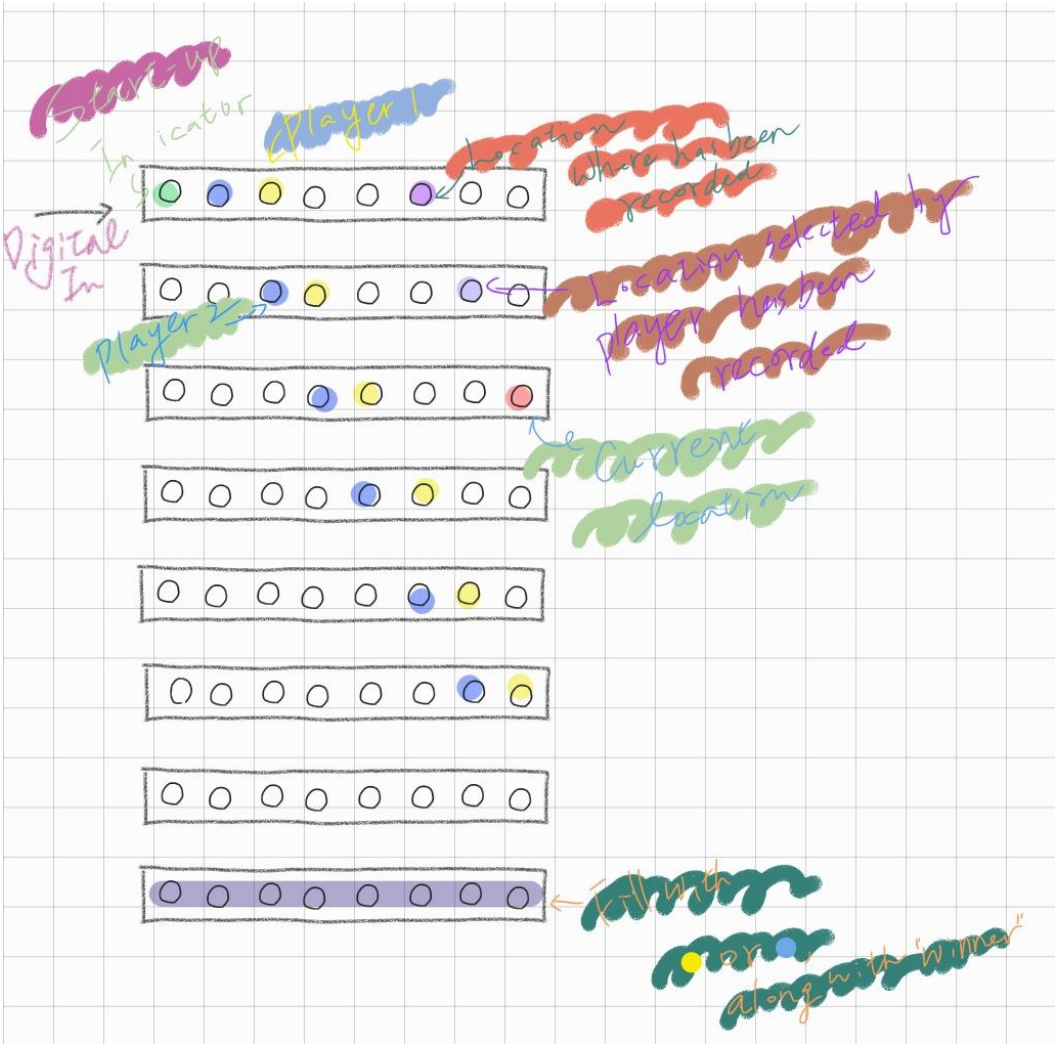

圖 2-1 成品構想與構圖

（使用 Samsung Notes 繪成）

壹、構思說明：

（一）程式初次執行或遊戲重新開始準備已就緒時，左上角亮綠燈。

（二）代表玩家一的顏色為「黃色」，玩家二為「藍色」。

（三）遊戲中已被紀錄的位置為兩玩家之各自代表色。

（四）指標目前所在的位置為「紅色」，玩家可以藉由按鈕控制指標移動。

（五）當指標所在之位置已有玩家紀錄，該位置指示燈變為「紫色」。

（六）承（五），此時若玩家按下確認鈕，該位置之指示燈變為「淡紫色」，且行動不被記錄，即棋盤沒有更動且玩家沒有交替。

（七）承（五）、（六），此時若使用方向控制鈕從該位置將指標移開，原位置將變回原紀錄之玩家代表色，指標仍呈現「紅色」。

（八）當有玩家勝出，最底下整排呈現贏家之代表色，並於清除紀錄後，最底排持續呈現上述顏色，並亮起左上角指示燈，為「綠色」。

第二節　電路部分

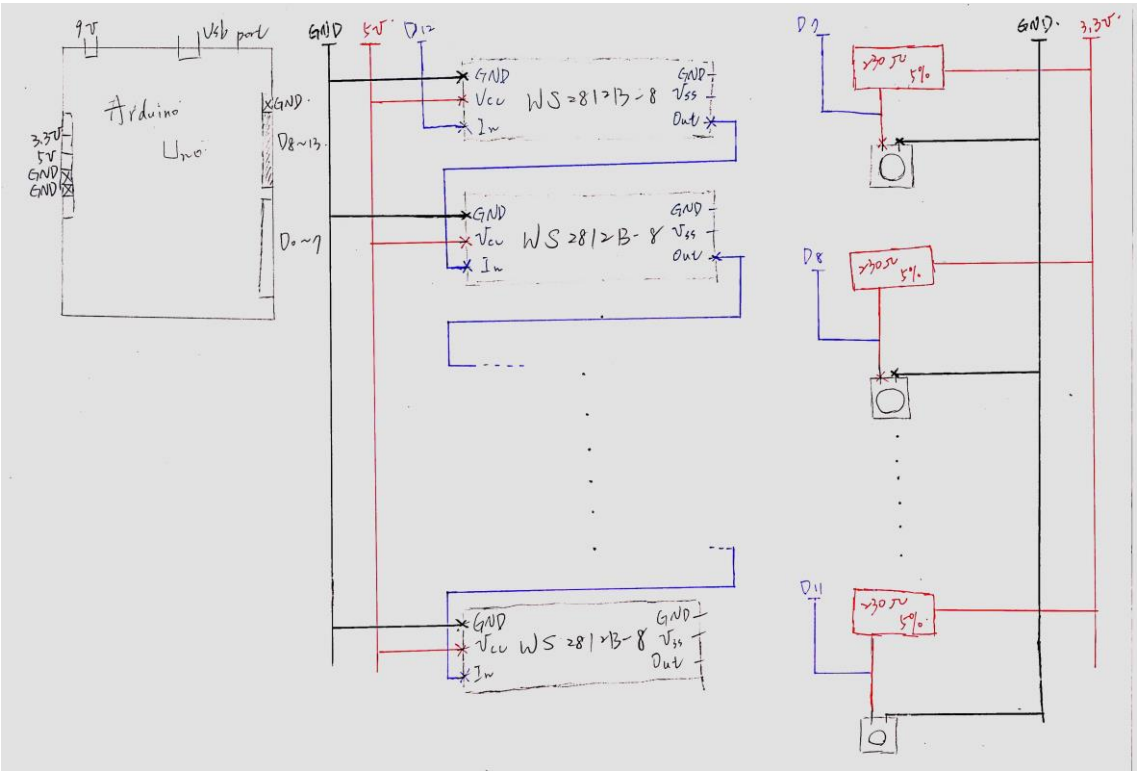

圖 2-2 電路構想與構圖

壹、上拉電阻 (Pull-up resistors)

如圖 2-2 所示，右半部採用上拉電阻的結構，用以控制棋盤上之指標移動或進行確認的動作。

以第一顆按鈕為例，當按鈕沒有被按下時，D7 訊號線傳給 Arduino 的邏輯電平為 [High]，反之，Arduino 將收到 [Low] 的訊號。藉由此訊號控制指標的動作。

貳、WS2812-B 模組間的電路安排及訊號連接

如圖 2-2 所示，左半部為「棋盤」，各燈條模組使用並聯共同使用一個輸入電源及接地，第一個模組先以 D12 腳位被輸入訊號，而後皆透過尾端 Dout 接點傳遞訊號給下一模組之 Din，線路連接構想參考下圖。

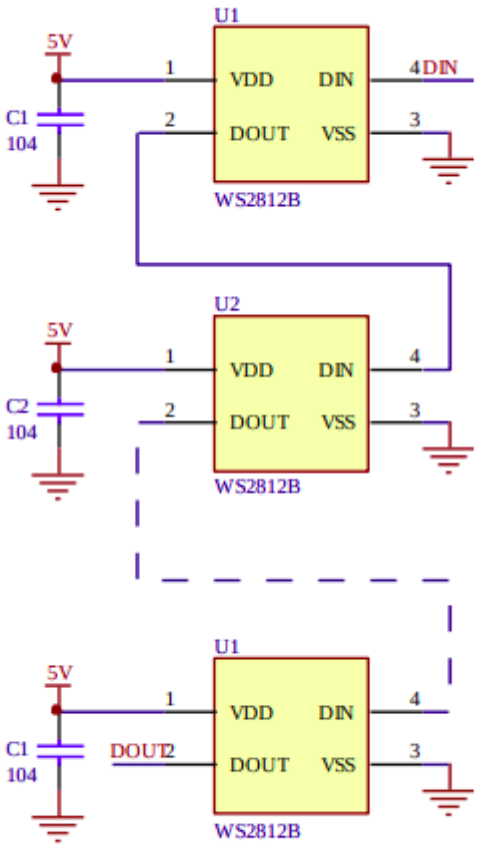

圖 2-3 WS2812-B 電路參考圖 [4]

資料來源：Worldsemi / WS2812-B reference

圖 2-3 中，接地方式是使用 Vss 接點做共地，而圖 2-2 中則是將 Vss 替換成 GND 接點，兩者於本實驗裝置中並無太大的差異，對裝置不會造成影響。

第三節 程式部分

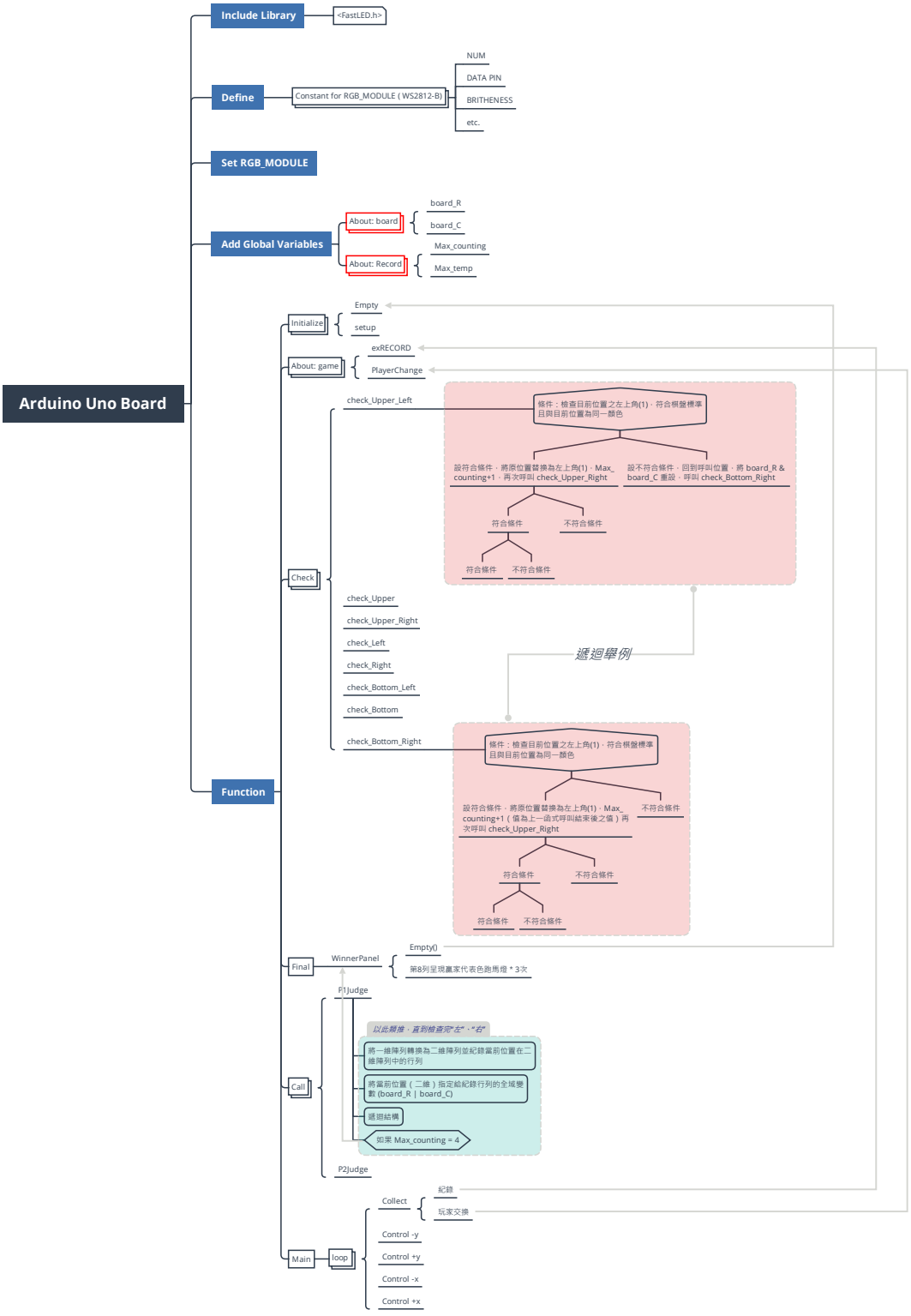

圖 2-4 程式構想

（使用 XMind 繪成）

第三章　實作與測試

第一節　電路部分

壹、WS2812-B 模組

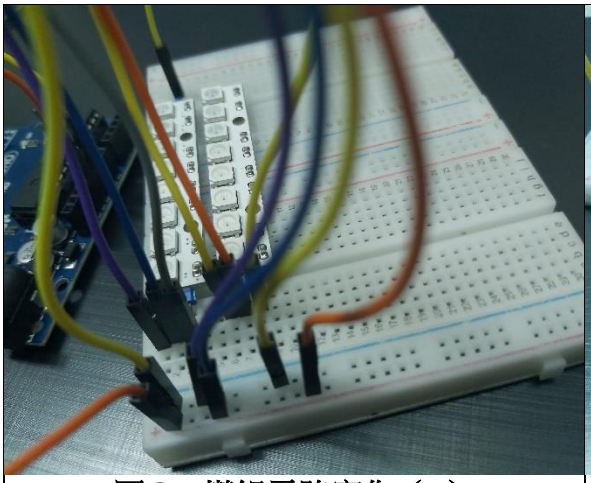

圖 3-1 模組電路實作（1）

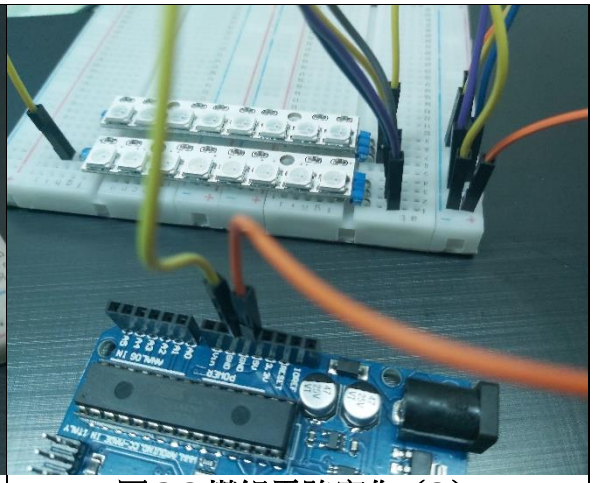

圖 3-2 模組電路實作（2）

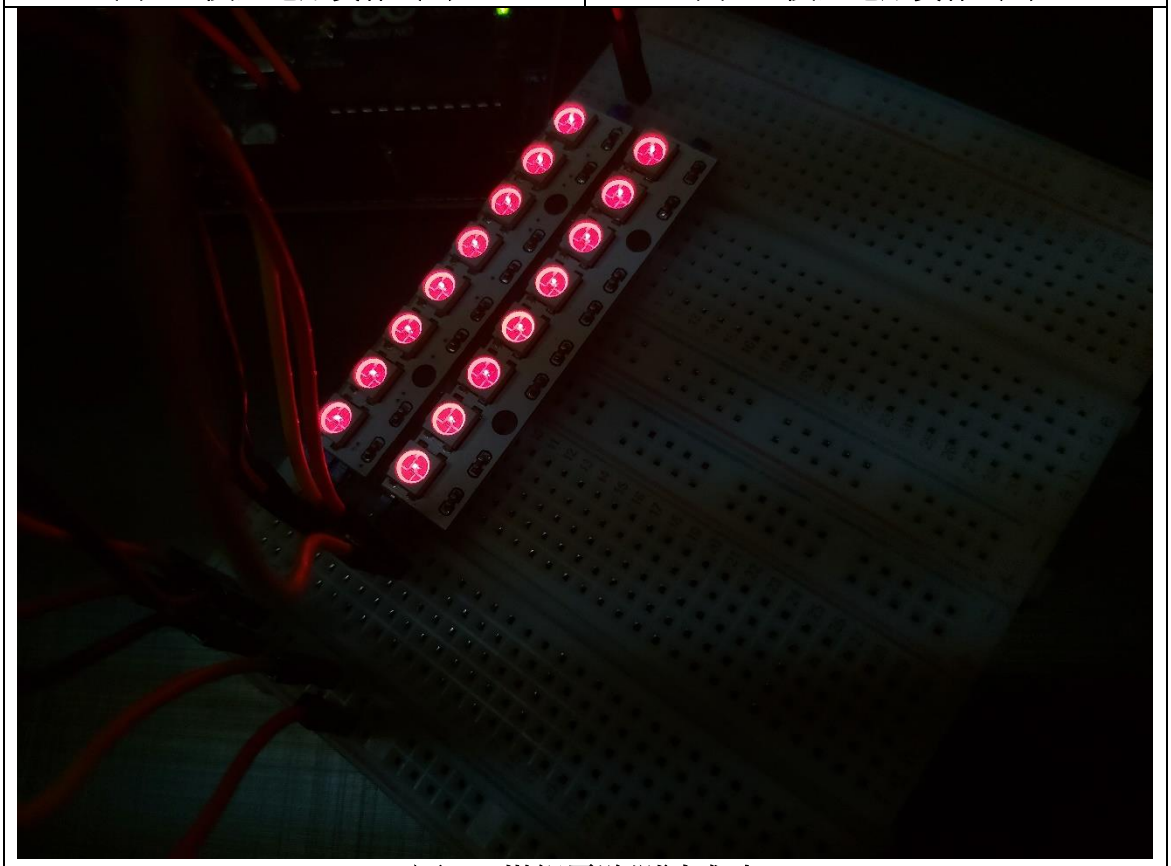

圖 3-3 模組電路測試成功

貳、上拉電阻按鈕

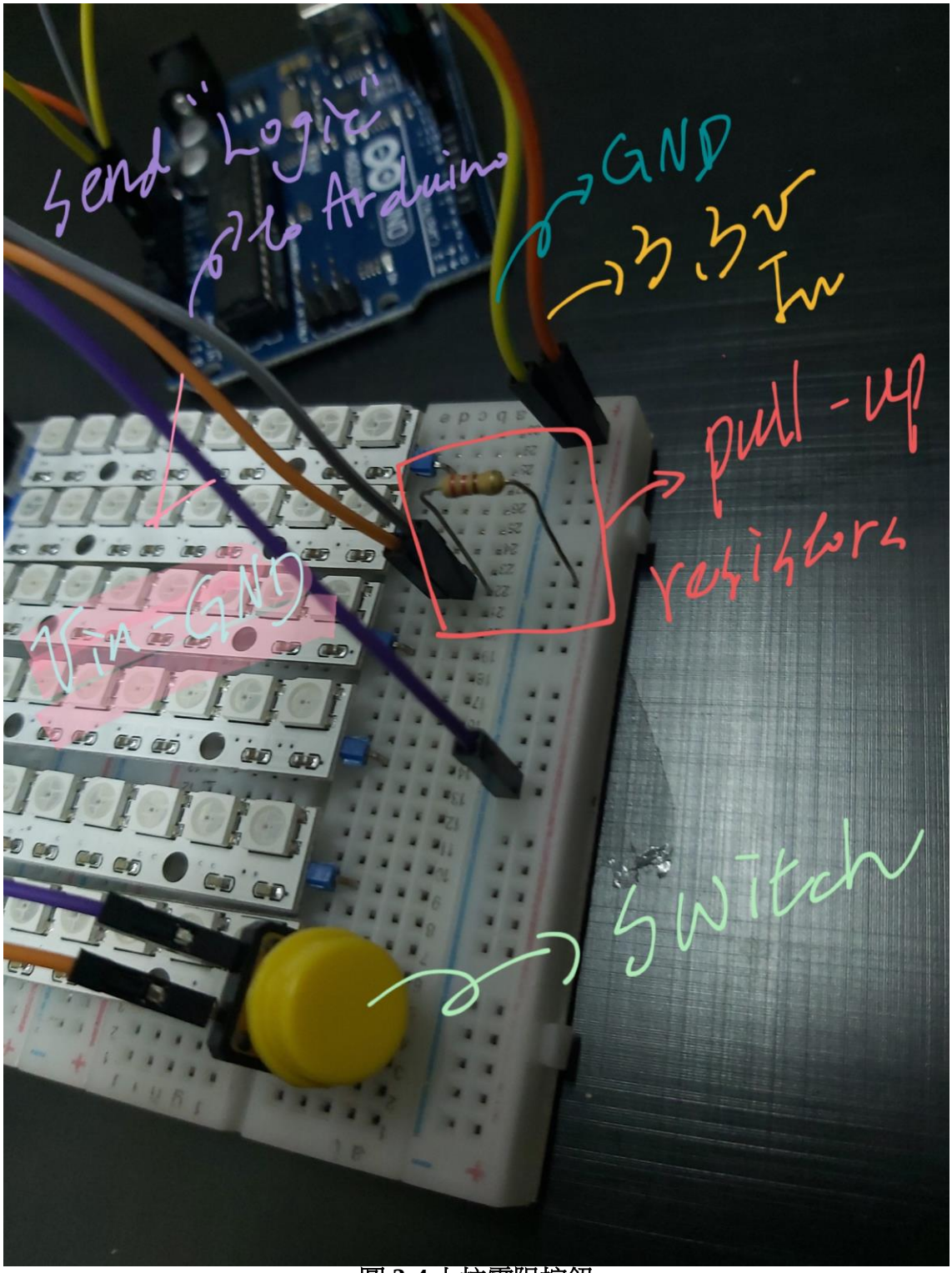

圖 3-4 上拉電阻按鈕

第二節 程式部分

（以下程式碼全為 丁文瑩 自行編寫完成，並無使用其他資源取得部分或全部之程式碼或架構，FastLED 函式庫相關除外）

```
#include <FastLED.h>

#define RGB_NUM 48
#define RGB_DATA_PIN 12
#define RGB_BRITHENESS 10
#define RGB_MODEL WS2812B
#define RGB_ORDER GRB
#define RGB_COLUMN 8
#define RGB_ROW 6
#define Wait 300
#define RB_DATA_PIN 6

CRGB RGB_MODULE[RGB_NUM];

int RECORD[48] = {0};
int board[6][8] = {0};
int LOCATION = 0;
int preLOCATION = 0;
int CurPlayer = 1;

/*--------------set for check_panel--------------*/

int drLocation = 0;
int Max_counting = 0;
int board_R = RGB_ROW / 2;
int board_C = RGB_COLUMN / 2;
int temp_R = 0, temp_C = 0;
/*--------------set for check_panel--------------*/

void Empty()                                    //清空紀錄及版面
{
    for (int i = 0; i < 48; i++)
        RECORD[i] = 0;
    for (int i = 0; i < 6; i++)
        for (int r = 0; r < 8; r++)
            board[i][r] = 0;
    for (int i = 0; i < 49; i++)
        RGB_MODULE[i] = 0;

    LOCATION = 0;
    CurPlayer = 1;

    RGB_MODULE[0] = CRGB::Green;
    FastLED.show();
}

void setup()
{
    LEDS.addLeds<RGB_MODEL, RGB_DATA_PIN, RGB_ORDER>(RGB_MODULE, RGB_NUM);
    FastLED.setBrightness(RGB_BRITHENESS);

    pinMode(11, INPUT);
    pinMode(10, INPUT);
    pinMode(9, INPUT);
    pinMode(8, INPUT);
    pinMode(7, INPUT);

    Empty();
}

void exRECORD(int location, int player)
{
    RECORD[location] = player;
    for (int i = 0; i < 48; i++)
    {
        if (RECORD[i] == 1)
            RGB_MODULE[i] = CRGB::Yellow;
        if (RECORD[i] == 2)
            RGB_MODULE[i] = CRGB::Blue;
    }
    FastLED.show();
}

int PlayerChange(int CurrentPlayer)
{
    if (CurrentPlayer == 1)
        return 2;
    if (CurrentPlayer == 2)
        return 1;
```

```
}

/*----------CHECK----------*/

void check_Upper_Left()
{
    if (board_R - 1 > -1 && board_C - 1 > -1 && board[board_R][board_C] == board[board_R - 1][board_C - 1])
    {
        board_R -= 1;
        board_C -= 1;
        Max_counting += 1;
        check_Upper_Left();          //如果滿足上述條件，重複呼叫
    }
}

void check_Upper()
{
    if (board_R - 1 > -1 && board[board_R][board_C] == board[board_R - 1][board_C])
    {
        board_R -= 1;
        Max_counting += 1;
        check_Upper();
    }
}

void check_Upper_Right()
{
    if (board_R - 1 > -1 && board_C + 1 < 8 && board[board_R][board_C] == board[board_R - 1][board_C + 1])
    {
        board_R -= 1;
        board_C += 1;
        Max_counting += 1;
        check_Upper_Right();
    }
}

void check_Left()
{
    if (board_C - 1 > -1 && board[board_R][board_C] == board[board_R][board_C - 1])
    {
        board_C -= 1;
        Max_counting += 1;
        check_Left();
    }
}

void check_Right()
{
    if (board_C + 1 < 8 && board[board_R][board_C] == board[board_R][board_C + 1])
    {
        board_C += 1;
        Max_counting += 1;
        check_Right();
    }
}

void check_Bottom_Left()
{
    if (board_R + 1 < 6 && board_C - 1 > -1 && board[board_R][board_C] == board[board_R + 1][board_C - 1])
    {
        board_R += 1;
        board_C -= 1;
        Max_counting += 1;
        check_Bottom_Left();
    }
}

void check_Bottom()
{
    if (board_R + 1 < 6 && board[board_R][board_C] == board[board_R + 1][board_C])
    {
        board_R += 1;
        Max_counting += 1;
        check_Bottom();
    }
}

void check_Bottom_Right()
{
```

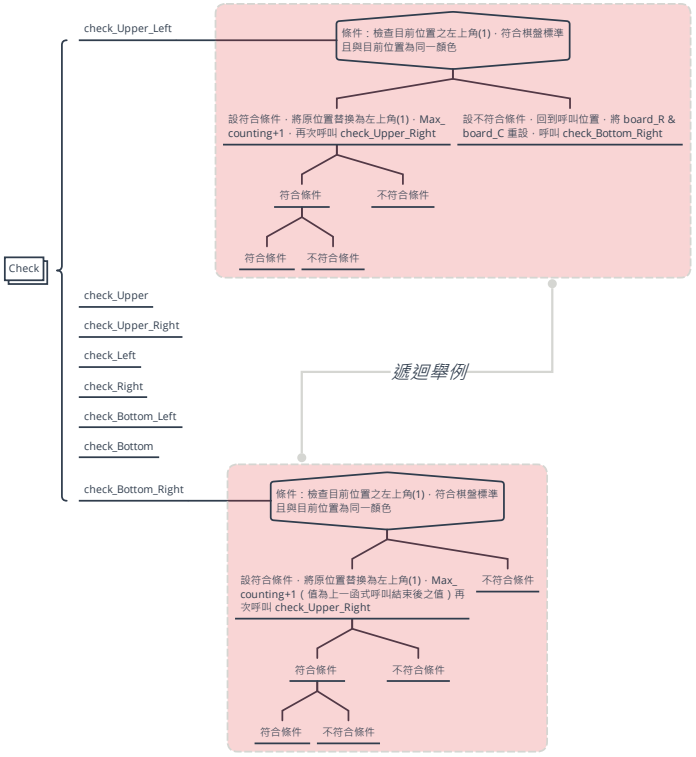

圖 3-5 About: Check

（使用 XMind 繪成）

```
159.        if (board_R + 1 < 6 && board_C + 1 < 8 && board[board_R][board_C] == board[board_R + 1][board_C + 1])
160.        {
161.            board_R += 1;
162.            board_C += 1;
163.            Max_counting += 1;
164.            check_Bottom_Right();
165.        }
166.    }
167.
168.    /*-----------------------------------------------CHECK-----------------------------------------------*/
169.
170.    void WinnerPanel(int winner)
171.    {
172.        Empty();
173.        if (winner == 1)
174.            for (int r = 0; r < 3; r++)
175.            {
176.                for (int i = 40; i < 48; i++)
177.                {
178.                    RGB_MODULE[i] = CRGB::Yellow;
179.                    FastLED.show();
180.                    delay(50);
181.                    RGB_MODULE[i] = 0;
182.                    FastLED.show();
183.                }
184.            }
185.        if (winner == 2)
186.            for (int r = 0; r < 3; r++)
187.            {
188.                for (int i = 40; i < 48; i++)
189.                {
190.                    RGB_MODULE[i] = CRGB::Blue;
191.                    FastLED.show();
192.                    delay(50);
193.                    RGB_MODULE[i] = 0;
194.                    FastLED.show();
195.                }
196.            }
197.        Empty();
198.    }
199.
200.    void P1Judge(int location)
201.    {
202.        for (int r = 0; r < RGB_ROW; r++)
203.            for (int c = 0; c < RGB_COLUMN; c++)
204.            {
205.                board[r][c] = RECORD[drLocation];
206.                if (drLocation == location)
207.                {
208.                    temp_R = r;
209.                    temp_C = c;
210.                }
211.                drLocation++;
212.            }
213.        drLocation = 0;
214.
215.        board_R = temp_R;
216.        board_C = temp_C;
217.        check_Upper_Left();
218.        board_R = temp_R;
219.        board_C = temp_C;
220.        check_Bottom_Right();
221.        if (Max_counting == 4)
222.            WinnerPanel(board[board_R][board_C]);
223.        Max_counting = 0;
224.
225.        board_R = temp_R;
226.        board_C = temp_C;
```

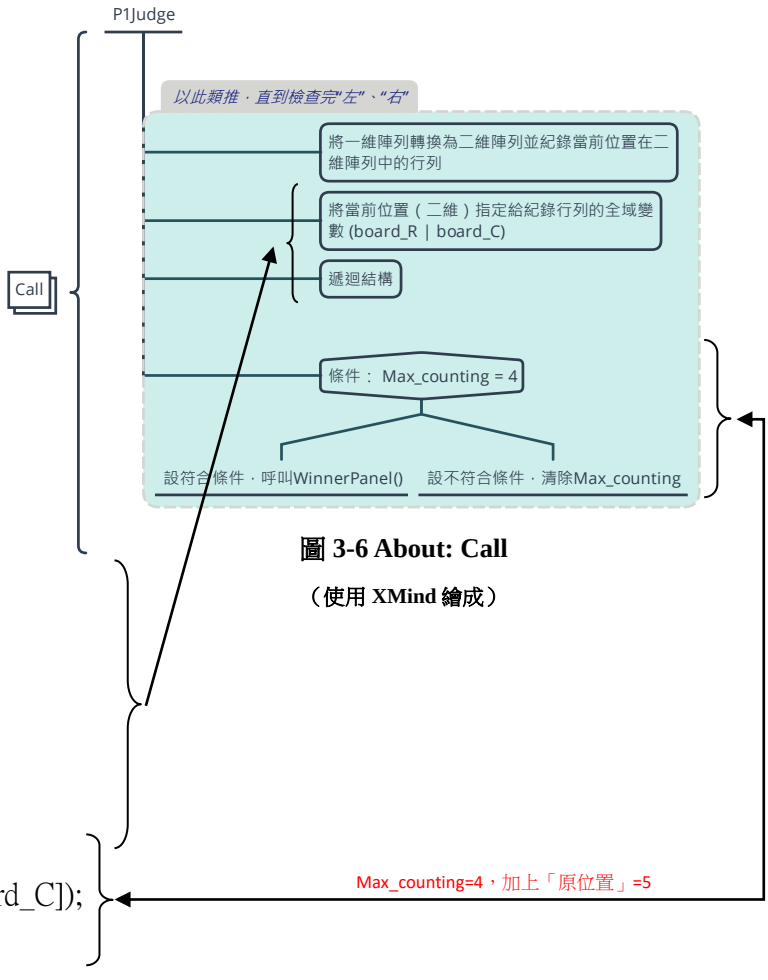

圖 3-6 About: Call

（使用 XMind 繪成）

```
227.        check_Upper();
228.        board_R = temp_R;
229.        board_C = temp_C;
230.        check_Bottom();
231.        if (Max_counting == 4)
232.            WinnerPanel(board[board_R][board_C]);
233.        Max_counting = 0;
234.
235.        board_R = temp_R;
236.        board_C = temp_C;
237.        check_Upper_Right();
238.        board_R = temp_R;
239.        board_C = temp_C;
240.        check_Bottom_Left();
241.        if (Max_counting == 4)
242.            WinnerPanel(board[board_R][board_C]);
243.        Max_counting = 0;
244.
245.        board_R = temp_R;
246.        board_C = temp_C;
247.        check_Left();
248.        board_R = temp_R;
249.        board_C = temp_C;
250.        check_Right();
251.        if (Max_counting == 4)
252.            WinnerPanel(board[board_R][board_C]);
253.        Max_counting = 0;
254.    }
255.
256.    void P2Judge(int location)
257.    {
258.        for (int r = 0; r < RGB_ROW; r++)
259.            for (int c = 0; c < RGB_COLUMN; c++)
260.            {
261.                board[r][c] = RECORD[drLocation];
262.                if (drLocation == location)
263.                {
264.                    temp_R = r;
265.                    temp_C = c;
266.                }
267.                drLocation++;
268.            }
269.        drLocation = 0;
270.
271.        board_R = temp_R;
272.        board_C = temp_C;
273.        check_Upper_Left();
274.        board_R = temp_R;
275.        board_C = temp_C;
276.        check_Bottom_Right();
277.        if (Max_counting == 4)
278.            WinnerPanel(board[board_R][board_C]);
279.        Max_counting = 0;
280.
281.        board_R = temp_R;
282.        board_C = temp_C;
283.        check_Upper();
284.        board_R = temp_R;
285.        board_C = temp_C;
286.        check_Bottom();
287.        if (Max_counting == 4)
288.            WinnerPanel(board[board_R][board_C]);
289.        Max_counting = 0;
290.
291.        board_R = temp_R;
292.        board_C = temp_C;
293.        check_Upper_Right();
294.        board_R = temp_R;
295.        board_C = temp_C;
296.        check_Bottom_Left();
```

```
297.        if (Max_counting == 4)
298.            WinnerPanel(board[board_R][board_C]);
299.        Max_counting = 0;
300.
301.        board_R = temp_R;
302.        board_C = temp_C;
303.        check_Left();
304.        board_R = temp_R;
305.        board_C = temp_C;
306.        check_Right();
307.        if (Max_counting == 4)
308.            WinnerPanel(board[board_R][board_C]);
309.        Max_counting = 0;
310.    }
311.
312.    void loop()
313.    {
314.        /*COLLECT*/
315.        if (digitalRead(11) == LOW)
316.        {
317.            if (RECORD[LOCATION] != 0)
318.            {
319.                RGB_MODULE[LOCATION] = CRGB::Violet;
320.                FastLED.show();
321.                delay(Wait);
322.            }
323.            else
324.            {
325.                exRECORD(LOCATION, CurPlayer);  //紀錄
326.                if (CurPlayer == 1)
327.                    P1Judge(LOCATION);
328.                else
329.                    P2Judge(LOCATION);
330.                CurPlayer = PlayerChange(CurPlayer);  //玩家交換
331.                delay(Wait);
332.            }
333.        }
334.        /*Stage Complete*/
335.
336.        /*----------Direction Control----------*/
337.        /*Control -Y*/
338.        if (digitalRead(8) == LOW)
339.            if (LOCATION <= 39)
340.            {
341.                preLOCATION = LOCATION;
342.                LOCATION += 8;
343.                if (RECORD[LOCATION] != 0)
344.                    RGB_MODULE[LOCATION] = CRGB::Purple;
345.                else
346.                    RGB_MODULE[LOCATION] = CRGB::Red;
347.                if (RECORD[preLOCATION] != 0)
348.                {
349.                    if (RECORD[preLOCATION] == 1)
350.                        RGB_MODULE[preLOCATION] = CRGB::Yellow;
351.                    if (RECORD[preLOCATION] == 2)
352.                        RGB_MODULE[preLOCATION] = CRGB::Blue;
353.                }
354.                if (RECORD[preLOCATION] == 0)
355.                    RGB_MODULE[preLOCATION] = 0;
356.                FastLED.show();
357.                delay(Wait);
358.            }
359.        /*Stage Complete*/
360.
361.        /*Control +Y*/
362.        if (digitalRead(7) == LOW)
363.            if (LOCATION >= 8)
364.            {
365.                preLOCATION = LOCATION;
366.                LOCATION -= 8;
367.                if (RECORD[LOCATION] != 0)
368.                    RGB_MODULE[LOCATION] = CRGB::Purple;
369.                else
370.                    RGB_MODULE[LOCATION] = CRGB::Red;
371.                if (RECORD[preLOCATION] != 0)
```

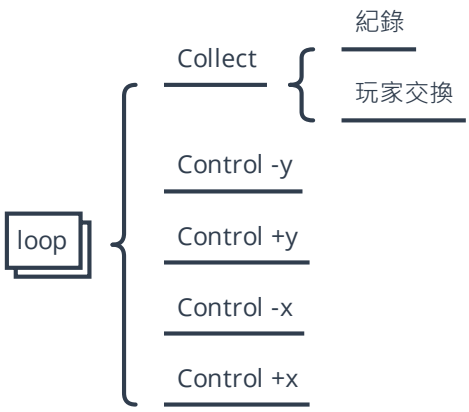

圖 3-7 About: loop

（使用 XMind 繪成）

```
                {
                    if (RECORD[preLOCATION] == 1)
                        RGB_MODULE[preLOCATION] = CRGB::Yellow;
                    if (RECORD[preLOCATION] == 2)
                        RGB_MODULE[preLOCATION] = CRGB::Blue;
                }
                if (RECORD[preLOCATION] == 0)
                    RGB_MODULE[preLOCATION] = 0;
                FastLED.show();
                delay(Wait);
            }
        /*Stage Complete*/

        /*Control -x*/
        if (digitalRead(10) == LOW)
            if (LOCATION != 0 && LOCATION != 8 && LOCATION != 16 && LOCATION != 24 && LOCATION != 32 && LOCATION != 40)
            {
                preLOCATION = LOCATION;
                LOCATION -= 1;
                if (RECORD[LOCATION] != 0)
                    RGB_MODULE[LOCATION] = CRGB::Purple;
                else
                    RGB_MODULE[LOCATION] = CRGB::Red;
                if (RECORD[preLOCATION] != 0)
                {
                    if (RECORD[preLOCATION] == 1)
                        RGB_MODULE[preLOCATION] = CRGB::Yellow;
                    if (RECORD[preLOCATION] == 2)
                        RGB_MODULE[preLOCATION] = CRGB::Blue;
                }
                if (RECORD[preLOCATION] == 0)
                    RGB_MODULE[preLOCATION] = 0;
                FastLED.show();
                delay(Wait);
            }
        /*Stage Complete*/

        /*Control +x*/
        if (digitalRead(9) == LOW)
            if (LOCATION != 7 && LOCATION != 15 && LOCATION != 23 && LOCATION != 31 && LOCATION != 39 && LOCATION != 48)
            {
                preLOCATION = LOCATION;
                LOCATION += 1;
                if (RECORD[LOCATION] != 0)
                    RGB_MODULE[LOCATION] = CRGB::Purple;
                else
                    RGB_MODULE[LOCATION] = CRGB::Red;
                if (RECORD[preLOCATION] != 0)
                {
                    if (RECORD[preLOCATION] == 1)
                        RGB_MODULE[preLOCATION] = CRGB::Yellow;
                    if (RECORD[preLOCATION] == 2)
                        RGB_MODULE[preLOCATION] = CRGB::Blue;
                }
                if (RECORD[preLOCATION] == 0)
                    RGB_MODULE[preLOCATION] = 0;
                FastLED.show();
                delay(Wait);
            }
        /*Stage Complete*/

        /*----------Direction Control----------*/
    }
```

第四章　展示

影片連結：

https://drive.google.com/file/d/1sRXGyoOSHdLlKZ2Ip_Llhev5yt0IrWEn/view?usp=sharing

程式碼版本控制：

Http: https://github.com/wynne-c-prog/Arduino-gobang.git

Ssh: git@github.com:wynne-c-prog/Arduino-gobang.git

第五章　研究心得

丁文瑩：

這次研究的目的，主要是想精進程式設計中「遞迴邏輯」的能力，因緣際會下想到了棋盤搜尋與之可能的關聯性，因而設計了此次研究。研究期間，曾困擾於如何判定各玩家下的每一步棋是否有達成五子棋勝利的條件，在幾次重新思考及實驗後修改了程式碼，終於可以成功辨別每一種可以勝利的下法。以下以第一、二次的程式版本為例。

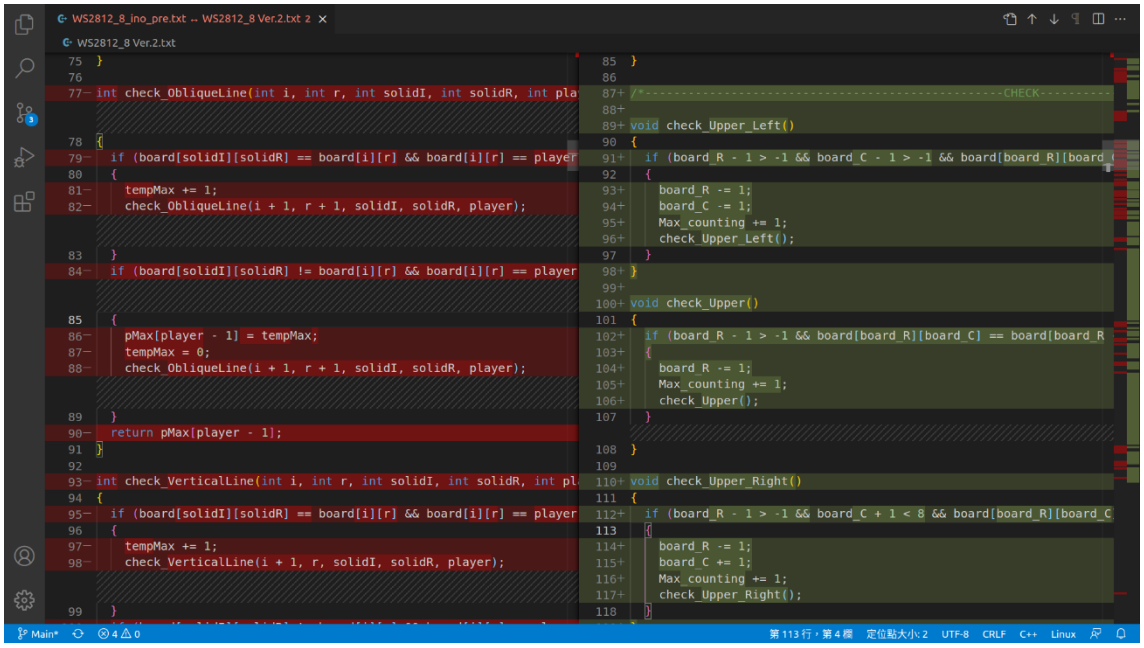

圖 5-1 程式版本差異舉例

如圖 5-1，左半部為初版，右半部為第二版本。初版中 77-91 行為檢查斜線上棋的排列裝況，當初沒有想到有「左上-右下」與「右上-左下」兩個方向，設計方式複雜且沒有良好的運用到遞迴；反觀第二版本，將「斜線」部分切割成四個部分（89-98 行，以「左上」為例），雖然在呼叫時多了許多步驟，但是各函式分工清楚且使用遞迴的方式不斷縮小問題，於退出遞迴後與對應的函式之呼叫結果做加總並判斷。

經過這次研究，不但精進了關於遞迴結構的設計，也於預期外的學習了 Git 與 GitHub 的基礎觀念及其在終端機中使用的方法，雖然目前尚不熟練且時常遇到障礙，但相信之後會增進這方面的能力，以讓往後的程式開發更有效率且方便管理。

吳昱霖：

我是參與美工設計的部分，關於這部分，我們只是做了簡單的線路整理，為了盡量讓整個麵包板可以看起來不那麼亂，當我們線路接錯時可以更有效率去找尋是哪裡出問題。一開始剛用的時候當然會一直搞錯，但失敗了幾次後有比較熟悉了，綁完那些杜邦線後看著看著也蠻有成就感的。在整理的過程中，我們也請教過對 Arduino 這方面最有研究的組長，他也給過我們相當多的建議，也示範過怎麼給綁會比較好，但他也說，他只會跟我們講個大概，剩下的我們得要自己發掘，但有問題都可以請教他，要不是沒有組長教我們，我們可能會綁得亂七八糟吧。

李苡瑄：

這次的 Arduino 專題實作，是我第一次接觸到 Arduino 程式設計，目前有接觸電路學是在國三課程，現在終於有學以致用的感覺了。這半年每個禮拜兩節課都學習新的東西，才讓我們有現在這個成果。

那由於我第一次學習到 Arduino 程式設計，經由觀摩同學所設計該主題的程式，也讓我收穫很多。在討論過程中會有或多或少的摩擦、爭執，但我們彼此會產生體諒的默契，也更能加深團隊合作的精神，對於自己不熟悉的事物要如何分配工作也是這次進步的一部分。尤其因為疫情，為了避免感染疑慮，有些材料只能由特定的同學收取並且製作，雖然會有產生不平衡及不自信的想法，但經由調適及團隊們的溝通與鼓勵，大家又能繼續保持團結的氣氛，努力把這次的成果做到最好。

謝柏毅：

我這次是實作五子棋的實驗，我本身是修 Python 的，但有鑒於這次計畫使我對 C/C++有更高的認識，和同學討論除錯問題和想法很有收穫。

王柏崴：

本次 Arduino 專題研究我和吳昱霖同學主要負責電路的美工設計，而電路的設計和程式設計則是由丁文瑩同學一手包辦，成品出爐後又我負責美工，現實和

理想總有差距，原本預計美工是一件很簡單輕鬆的事，實際下去操作時才發現許多問題。

美工排版是否影響到程式進行是我們遇到的最大難關，若是美工設計影響到最重要的電路運行的話，那這樣大家辛苦製作出來的成品也是白費功夫，因此我們只好一步一步慢慢來，並且許多步驟都必須向電路和程式主要的負責人請教，許多排版是否會影響到實際的運作，而丁文瑩同學也很有耐心的和我們討論，最終一起完成了這項作品。

在這次的專題研究，講究的不是個人能力，而是合作的真諦，如果沒有大家的討論和分工，這項成品也沒有辦法如預期的順利完成，團結造就完美。

第六章　參考文獻

[1] 參見 Wikipedia, Arduino 條目

[2] 參考 打下好基礎—程式設計必修的數學思維與邏輯訓練（H&C 譯）（2016.08），周穎，台北：碁峰股份有限公司
ISBN: 9789864761401

[3] 使用 GitHub/FastLED
Http: https://github.com/FastLED/FastLED.git
Ssh: git@github.com:FastLED/FastLED.git

[4] 參考 WS2812B Reference (cdn shop)，P.5，Worldsemi

第七章　附錄

1.XMind 官方網站 https://www.xmind.net/

2.Git 參見 https://git-scm.com/ 或於 Terminal 中直接進行安裝

3.使用 GitHub 進行版本控制

Http: https://github.com/wynne-c-prog/Arduino-gobang.git

Ssh: git@github.com:wynne-c-prog/Arduino-gobang.git