# 2025年臺灣國際科學展覽會
# 優勝作品專輯

作品編號　190019

參展科別　電腦科學與資訊工程

作品名稱　基於高效可更新神經網絡的西洋棋人工智慧應用於嵌入式對弈棋盤

就讀學校　臺南市私立德光高級中學
指導教師　吳蓓怡
作者姓名　陳兆桉

關鍵詞　西洋棋、人工智慧、開關陣列

# 作者簡介


大家好，我是陳兆桉，目前就讀於德光高中的數理資優班二年級。我的興趣是下西洋棋，平時也喜歡學習各種新知識。今天非常榮幸能夠參與這屆國際科展。最初的動機來自於對棋盤設計的濃厚興趣，沒想到一做就是一年多。一路走來，我要特別感謝每一位在過程中支持和指導我的師長與同學。未來我也會繼續努力，精進自己的專業，迎接更多挑戰！

# 2025 年臺灣國際科學展覽會

## 研究報告

區別：

科別：電腦科學與資訊工程科

作品名稱：基於高效可更新神經網絡的西洋棋人工智慧應用於嵌入式對弈棋盤

關鍵詞：西洋棋、人工智慧、開關陣列

編號：

# 摘要

本研究以西洋棋為切入點，採用磁簧開關陣列來偵測棋子位置，並在設計中加入二極體以防止 Ghosting 效應，進而開發出一款以 Arduino Uno 開發板為基礎的智慧對弈棋盤。棋盤底部配備 RGB LED 燈，以便為使用者提供落子提示，並根據不同的落子類型呈現不同的燈光效果。

我們成功地透過簡潔的設計與高效的運算性能，實現了一個能夠識別棋手落子的智慧對弈棋盤，並能根據國際西洋棋規則提供正確的移動提示，讓完全沒有基礎的初學者也能在遊戲中學習並掌握西洋棋的所有規則。此外，我們還引入了基於 Minimax 演算法的輕量化 AI 和基於高效可更新神經網絡(NNUE)AI，並探討兩者之間的性能差異，從而使該智慧棋盤在節省運算資源的同時，可以在不連接電腦的前提下，具備一定的棋力，以支持棋手的技能提升與訓練。

# Abstract

This study uses chess as a starting point and employs a magnetic reed switch array to detect the position of chess pieces. Diodes are incorporated into the design to prevent ghosting effects, finally, Arduino UNO is used as the controller of the smart chessboard. The chessboard is equipped with RGB LED lights at the bottom to provide move prompts for users, displaying different lighting based on the type of move made.

Through a simple design and efficient computational performance, we successfully created a smart chessboard capable of recognizing player moves and providing correct move suggestions according to international chess rules. This allows complete beginners to learn and master all chess rules during gameplay. Additionally, with a lightweight AI which is based on the Minimax algorithm and an AI using Efficiently Updatable Neural Networks (NNUE), we explored the performance differences between the two. This enables the smart chessboard to offer a certain level of playing strength without being connected to a computer, while conserving computational resources, thus supporting players to improve their skill.

# 壹、 前言

## 一、研究動機

這次研究題目的想法，是我在教同學西洋棋時所發想的，新玩家在剛接觸西洋棋時，常因不熟悉規則，導致無從下手。這讓我不禁好奇，是否能設計出一款智能棋盤，能夠感應出棋手所持棋子，並提供輔助落子的指引，以達到輔導玩家對弈過程的目的。並且在無第二位玩家時，能夠提供一定棋力之 AI 對手，以達到訓練玩家棋力的作用。

## 二、研究目的

（一）研究需使用硬體並完成智能棋盤之設計。

（二）繪製棋盤結構設計圖並組裝成智慧棋盤。

（三）撰寫棋盤內部程式碼。

（四）實作 Minimax 演算法。

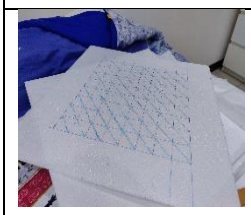
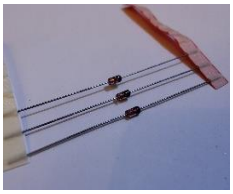
（五）實作高效可更新神經網絡(NNUE)。

（六）比較不同棋類演算法。

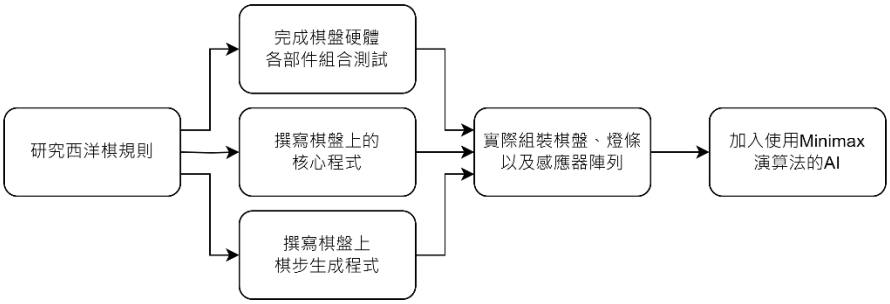
# 貳、 研究方法或過程

## 一、研究設備及器材

表一：研究設備與器材彙整表（資料來源：研究者自行製作）

| 珍珠板 | 杜邦線 | 二極體 | LED 發光燈條 | Arduino UNO 版 |
|---|---|---|---|---|
|  |  |  |  |  |
| 磁簧開關 | 磁吸式西洋棋 | 烙鐵 | 焊錫 | 3to8 解碼器 |
|  |  |  |  |  |
| 麵包板 | 筆記型電腦 | 尺 | 原子筆 | Arduino IDE |

二、研究流程圖

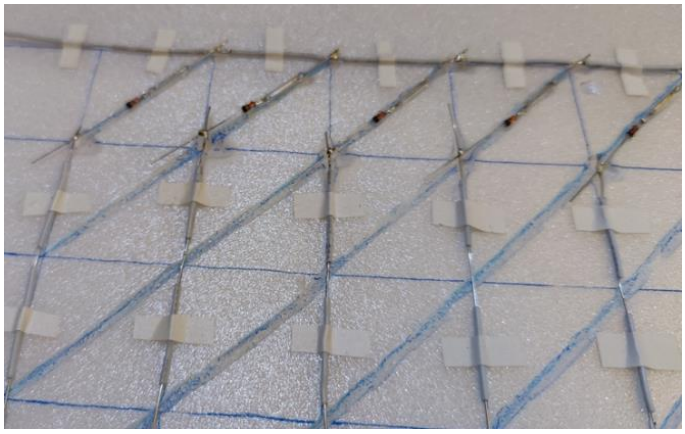



圖一：研究流程圖（資料來源：研究者自行繪製）

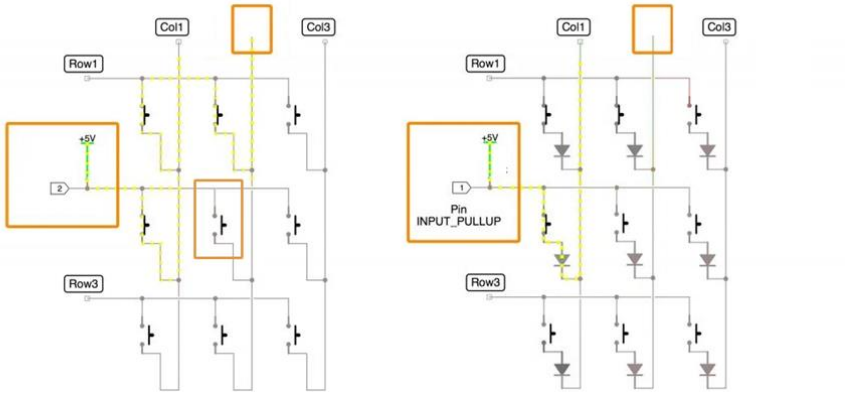
三、應用之硬體技術

（一）棋子偵測

原先我們打算使用霍爾感應器來偵測下方有嵌入磁石的西洋棋，但考慮到成本以及可行性，最後選擇使用磁簧開關代替，在最後的成品中，我們使用 8*8 總計 64 個磁簧開關來組成陣列（如下圖二），並以逐行掃描的方式偵測棋子在棋盤上的動態（詳見下圖三）。下圖二中與磁簧開關串連的原件為二極體，用來防止逐行掃描的偵測發生錯誤，類似的技術可以在鍵盤的開關陣列中看到，而前述掃描發生錯誤的成因就跟鍵盤中按鍵 Ghosting 的原因一樣，而加入二極體則可以解決該問題（如下圖三）。



圖二：磁簧開關陣列半成品圖（資料來源：研究者自行拍攝）




圖三：鍵盤 Ghosting 示意圖

（資料來源：https://www.baldengineer.com/arduino-keyboard-matrix-tutorial.html）

因為不能偵測棋子類型，所以我們需要持續追蹤場上的棋子位置，這樣就可以追蹤到整個棋盤的狀態，也就不需要讓棋盤能夠獨立判別每一顆棋子，是個比較可行且不會取捨掉太多功能的折衷方案。

而因為所使用的 Arduino UNO 開發板上面只有 13 個可以使用的腳位，若是要直接與整個磁簧開關陣列的線路對接會有腳位不夠的問題($8 \times 2 > 13$)，為了解決此問題我們利用了在每一次掃描中，橫向的八條線路只會需要有一條為高電位其他皆為低電位此特性，把原本在開發板上的八條輸出、八條輸入，利用在開發板以及開關陣列之間加入一個解碼器(Decoder)來將期簡化成三條輸出、八條輸入，讓開發板上還有其他空出的腳位可以連接其他東西，例如下方會介紹到的 LED 燈條。

（二）落子提示燈效

為了讓玩家能夠知道自己拿起來的棋子可以往哪些地方走，會需要有某種和玩家互動的方式，這時候我們就可以使用 LED 燈條來讓棋盤發亮（如下圖四）。

圖四：LED 燈條發光示意圖（資料來源：研究者自行拍攝）

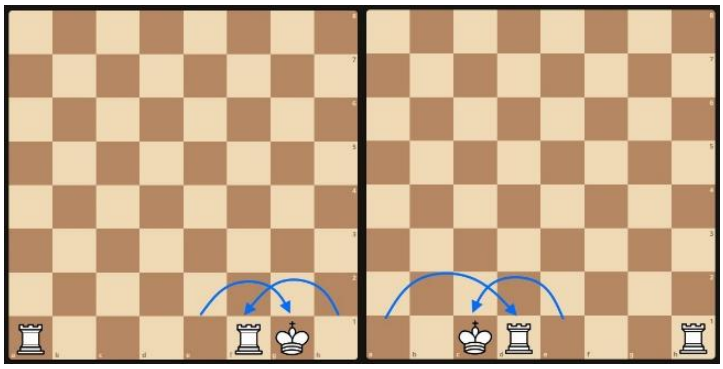
之所以選用 LED 燈條而不是 LED 陣列是因為 LED 陣列需要太多腳位以及線路，更不用說程式以及焊接了，但是若是使用 LED 燈條就可以一次解決這些問題，因為我們使用的 LED 燈條只需要三個腳位，正負極和 DataIn，只會占用開發板上面的一個 digital pin，除了需要的腳位很少以外，LED 燈條上每一顆燈珠都是由紅、綠和藍色三種顏色的 LED 燈所構成，也就是說，棋盤可以藉由不同燈光的顏色來代表不同的落子類型，例如單純的移動和吃子。

四、應用之軟體技術

（一）西洋棋基本規則

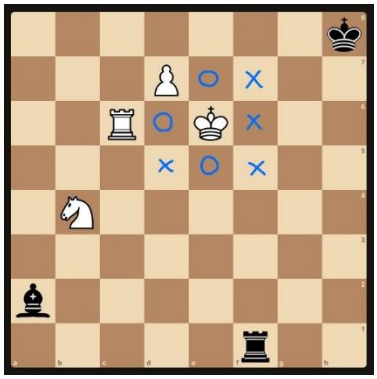
在能夠和機器人對弈之前，必須要先有基本規則，於是我們就撰寫了一套包含西洋棋所有基本規則判定的程式，並會列出棋盤上任意一棋子的合法棋步，接下來將展示程式實作的西洋棋規則。

1 國王入堡(Castling)：是指在國王以及城堡皆未移動過，且兩者之間並無其他棋子時，國王可以像城堡方向移動兩隔，並將城堡移動到另一側（如下圖五）。



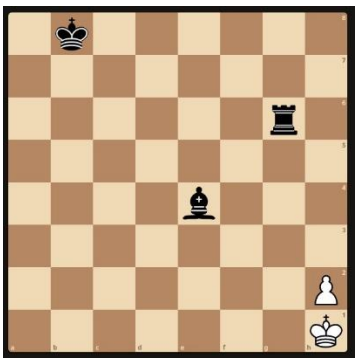
圖五：國王入堡示意圖（資料來源：研究者自行繪製）

2 將軍(Check)：是指國王遭到敵方棋子的攻擊，若是不處理，下一回合就會被敵方吃掉（如下圖六）。



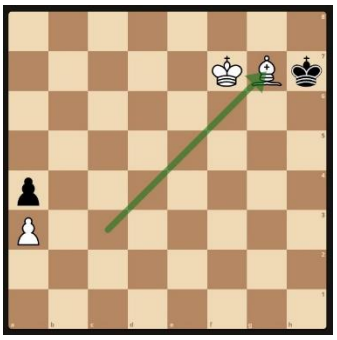
圖六：將軍示意圖（資料來源：研究者自行繪製）

3 將死(Check mate)：是指被將軍後不論如何處理，都無法化解掉的情況，若是被將死就等於輸棋（如下圖七）。



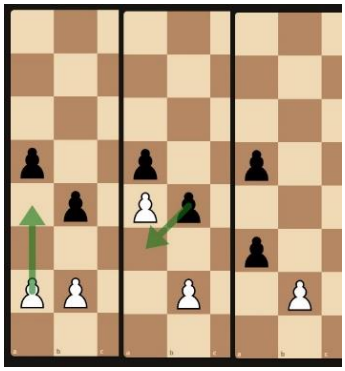
圖七：將死示意圖（資料來源：研究者自行繪製）

4　逼和(Stale mate)：是一種和局的情況，是指在要走棋的一方在國王沒有被將死的情況，所有棋子都沒有合法的走法（如下圖八）。



圖八：逼和示意圖（資料來源：研究者自行繪製）

5　吃過路兵(En passant)：此規則是在 15 世紀後才出現的，讓敵方的兵無法利用兵最初移動可以走兩隔的方法來避免該兵被我方的小兵吃掉，詳細走法（如下圖九）。



圖九：吃過路兵示意圖（資料來源：研究者自行繪製）

6　升變(Promotion)：升變是指當我方的兵成功走到敵方的最後一排時，可以選擇要變成自己的皇后、城堡、騎士、主教中的一種。

（二）棋盤掃描

程式除了要判斷並顯示可行的棋步以外，最重要的就是要如何正確的追蹤場上棋子的狀態，若是讓每一顆不同的棋子都可以在棋盤上的每一格被識別出來，

包含棋子類型，那所需要的硬體及軟體都會非常複雜，於使我們使用了只能判別棋盤上該位置有沒有棋子的解法，並利用了西洋棋每一場開局的棋子位置都是固定的這個特性，接著就只需要持續偵測哪裡的棋子被拿起，接著在哪裡放下，就可以持續追蹤棋盤上棋子的位置，但這也代表，我們必須要規定吃子時需要先拿起哪一方的棋子以避免程式判斷錯誤。

而根據 Decoder 的 Datasheet，從輸入傳遞到輸出並更新輸出的時間大約是 21 奈秒，而 16MHz 的 Arduino Uno 板，每個 Clock cycle 大約會花 62 奈秒，也就是說，因為處理器一個週期時間大於 Decoder 的更新時間，所以我們可以在執行掃描下一行棋盤的程式碼（更新 Decoder 輸入的程式碼）後面直接接著讀取掃描結果的程式碼，也不會有任何問題，非常方便。

（三）燈光互動

在玩家回饋方面本研究使用了 LED 燈條作為顯示合法棋步的媒介，因為其只使用到開發板上的一個 Digital pin，具體原理是因為在每一顆 LED 上都有一個小晶片（如下圖十），該晶片讓 LED 可以跟其他相同的 LED 串接在一起，並只用一條導線來通訊，LED 會把接收到的顏色訊號取一顆燈的資料下來之後把剩下的轉送給後面的 LED，例如有三個 LED，需要依序以「紅綠藍」亮起，那開發板就會依序發送「紅綠藍」的訊號，第一顆 LED 在接收到「紅綠藍」的訊號後，就會取第一個「紅」的訊號，並依照該顏色發亮，接著把剩下的「綠藍」訊號往後面的 LED 傳送，這樣一顆一顆傳下去就可以達到只利用一個 Data pin 就做到可以控制無限顆 LED 的功能。

圖十：RGB LED 燈珠中控制器示意圖
（資料來源：https://www.circuitgeeks.com/ws2812b-addressable-rgb-led-strip-with-arduino/）

而因為該 LED 燈條非常普遍，所以在 Arduino 的第三方函式庫中已經有支援該 LED 燈條的函式庫<FastLED.h>，該函式庫透過一個由其所提供的 CRGB 資料型態組成的陣列來控制 LED，要更改燈條上的顏色時，只需要更新陣列，並且呼叫 FastLED 的更新函式 FastLED.show()（如下圖十一），此函式庫把其他底層的東西抽象化掉，大大增加了程式的可讀性，也降低了開發的難度。

```
void loop() {
  // Turn the LED on
  leds[0] = CRGB::Red;
  leds[1] = CRGB::Green;
  leds[2] = CRGB::Blue;
  FastLED.show();
  delay(10);
}
```

圖十一：更新 LED 燈條之程式碼示意圖（資料來源：研究者自行製作）

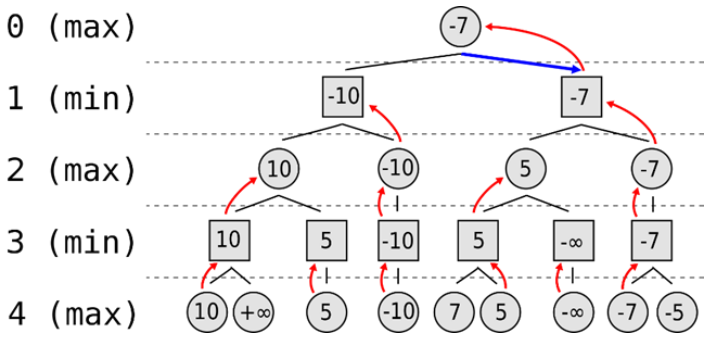
（四）對弈演算法

因為本研究是使用 Arduino UNO 來開發棋盤，基於其運算能力考量，我們使用並稍微修改了 H.G. Muller 製作的輕量化西洋棋 AI：Micro-Max 已讓他能夠在 Arduino UNO 上運行，該 AI 則是使用 Minimax 演算法來判斷下一步棋應該要落在哪裡最好，並且在文獻探討中的優化方式該 AI 都有實作出較簡單的版本，甚至還有想是寧靜搜索 (Quiescence search)、迭代深化(Iterative deepening)等，下方將會介紹幾個該 AI 使用到的核心演算法。

1 Minimax(Micro-Max 為了精簡則使用 NegaMax)

稱作極小化極大演算法，通常用在有完美資訊的雙人零和遊戲，例如圈圈叉叉，假設有 AB 兩個玩家要玩圈圈叉叉，Minimax 演算法會記錄一個局面的分數，最一開始局面分數會是 0，分數為正數代表對 A 方有利，負數則是對 B 方有利，於是玩家 A 就必須要將分數用的愈高愈好，玩家 B 則反之，換個角度講，兩方玩家就是要「『最小化』對方下一步可以獲得分數的『最大值』」，也就是該演算法名稱 Minimax 的由來。

接著舉例說明，下圖十二中每往下一層，代表遊戲中回合的增加，每一層都是由兩方玩家交替遊玩，也就是為什麼 min 以及 max 會交替出現的原因，通常 Minimax 會在往下展開整個遊戲樹到一定的深度後，再去衡量局面的分數，也就是圖中樹的最後一層，接著再慢慢往上選擇，min 層的玩家要盡量讓 max 的玩家選到比較低的分數，而 max 方則反之，也就是說，該演算法是在玩家雙方每一回合都使用最佳策略的前提之下去判斷最佳選擇的演算法。




圖十二：Minimax 演算法示意圖

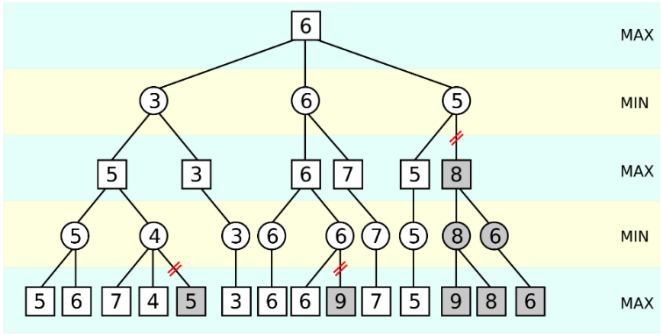
（資料來源：https://worksofr.wordpress.com/2019/07/05/connect-four-minimax-algorithm/）

但因為西洋棋的遊戲樹實在是太大，若是要全部展開，以目前家用電腦的效能來說是沒辦法的，因此並不能將遊戲樹展開到棋局結束，故需要有一個能夠在遊戲還沒有結束時能夠判斷整個局面對哪方玩家比較有利的審局函數，也就是說審局函數設計的好壞以及可以展開的深度會直接影響到 AI 的棋力，所以接著就來介紹一些優化 Minimax 演算法的方法。

## 2 Alpha-Beta 剪枝

是 Minimax 演算法的優化方式，名稱中的 Alpha 以及 Beta 取自演算法實作時，用來記錄目前最大化玩家選擇分數最大值的變數名稱 Alpha，以及用來記錄目前最小化玩家選擇分數最小值的變數名稱 Beta。

此演算法的核心概念為，以需要最小化局面分數的玩家為例，若已經知道最大化玩家上一步會走的分數最大值會是 6，那麼接著最小化玩家展開了 5 分的節點，那麼我們可以知道，不論接下來最小化玩家再展開什麼節點，都不會影響上一步最大化玩家的選擇。下圖十三為另外一個例子，途中展示了因為 Alpha-Beta 剪枝而被跳過的節點。




圖十三：Alpha-beta 剪枝示意圖
（資料來源：https://www.ericdrosado.com/2018/01/02/alpha-beta-pruning-in-minimax.html）

## 3 Transposition Table

又稱置換表，在棋盤上，常常會有用不同排列的棋步走到相同的局面的情況，但是若這種局面在之前就已經展開過了，再次展開的話會造成不必要的效能浪費，所以如果這時候可以辨識出已經展開過的局面，並利用查表的方式把展開的分數直接查出來，就可以省去重新展開該節點的時間了，以上就是 Transposition Table 的核心概念。

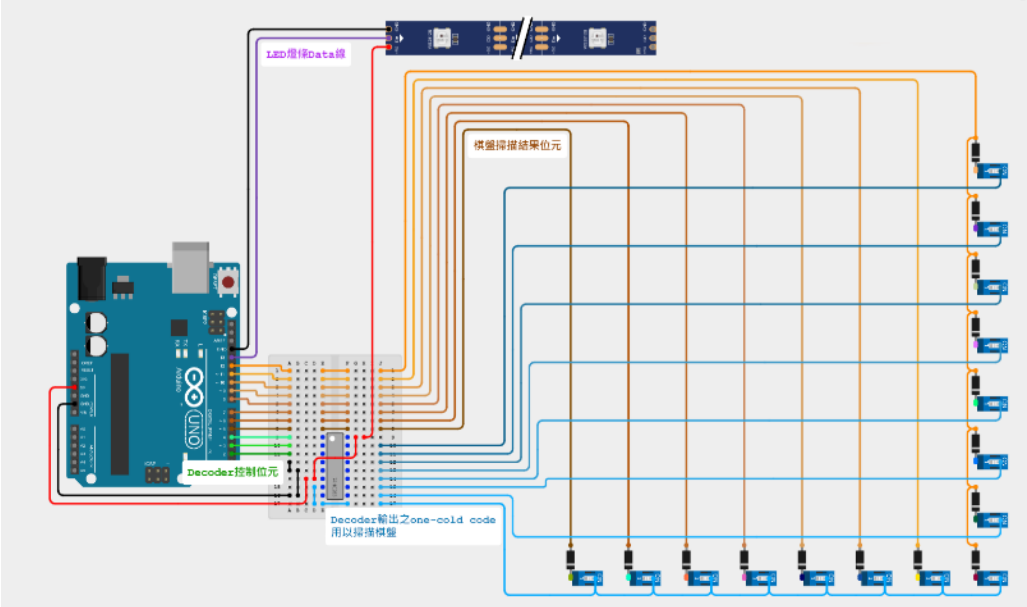
但是要有辦法快速的查表，就必須要有相對應的資料結構，通常是使用雜湊表(Hash Table)來實作，而使用到的雜湊函數則是 Zobrist Hash，此函數特別的地方在於，他會先對於棋盤上每一格的每一種可能的狀態（白方騎士、黑方國王或白方皇后等）都賦予一個隨機的數字（位元數相同），之後在計算整個局面的雜湊函數時只需要遍歷棋盤並將每格相對應該棋子狀態的隨機數 XOR 在一起就好，為什麼是使用 XOR 是因為除了第一次計算局面的雜湊

值以外，剩下所有回合的雜湊值都可以利用和前一回合的變動來更新，不需要再次遍歷整個棋盤，例如原本局面的雜湊值為 H，現在玩家把兵從 e2 移動到 e3，e2 上有小兵的隨機數為 R_e2，e3 上有小兵的隨機數為 R_e3，則這時候的雜湊值就會是 H⊕▒〖R_e2 ⊕▒R_e3 〗，主要是利用(A⊕B)⊕B=A 這個特性來先把 e2 的小兵清除，接著在 e3 放上小兵。也正是因為 Zobrist Hash 不需要從頭重新運算的特性，讓他被選為 Transposition Table 中的雜湊演算法。

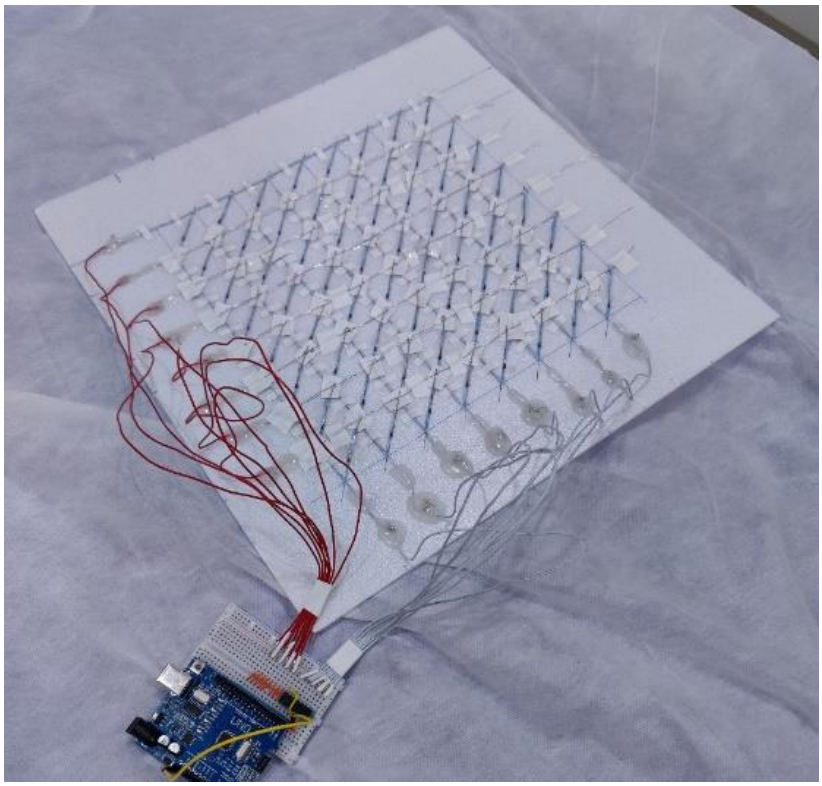
# 參、 研究結果與討論

## 一、智慧棋盤線路圖

下圖十四為我們整個智慧西洋棋盤的線路設計圖，右側的磁簧開關二極體陣列因若全部繪製出來將會使得版面過於混亂，故只繪製了各一排，雖然沒有繪製上去，但是圖中陣列每兩條線交界處都有一個磁簧開關以及二極體，並由總共 64 個此部件組成。

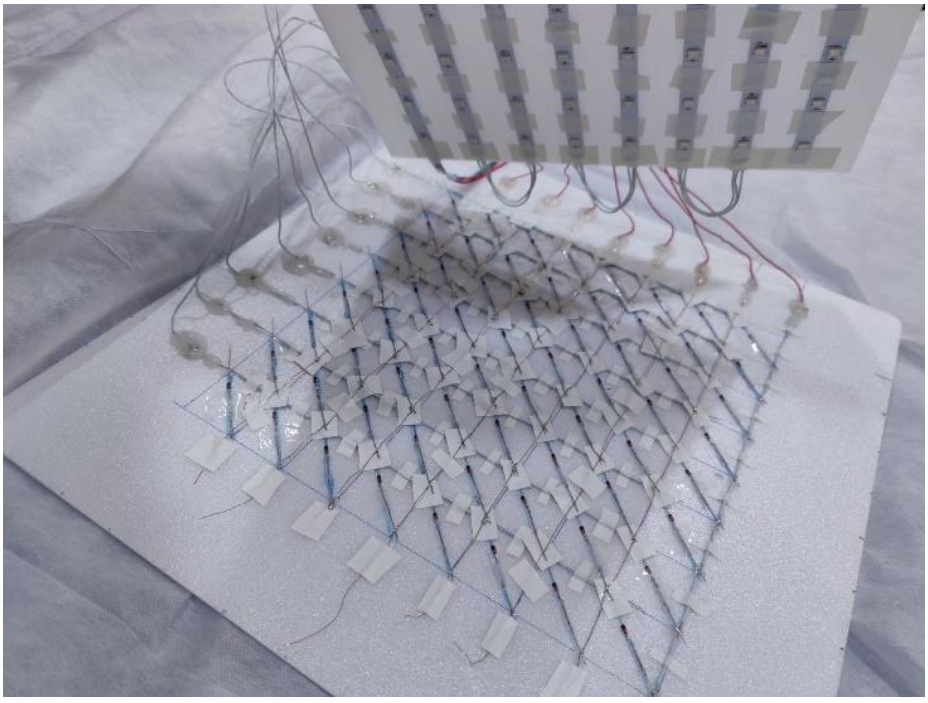



圖十四：智慧棋盤線路設計圖（資料來源：研究者自行繪製）



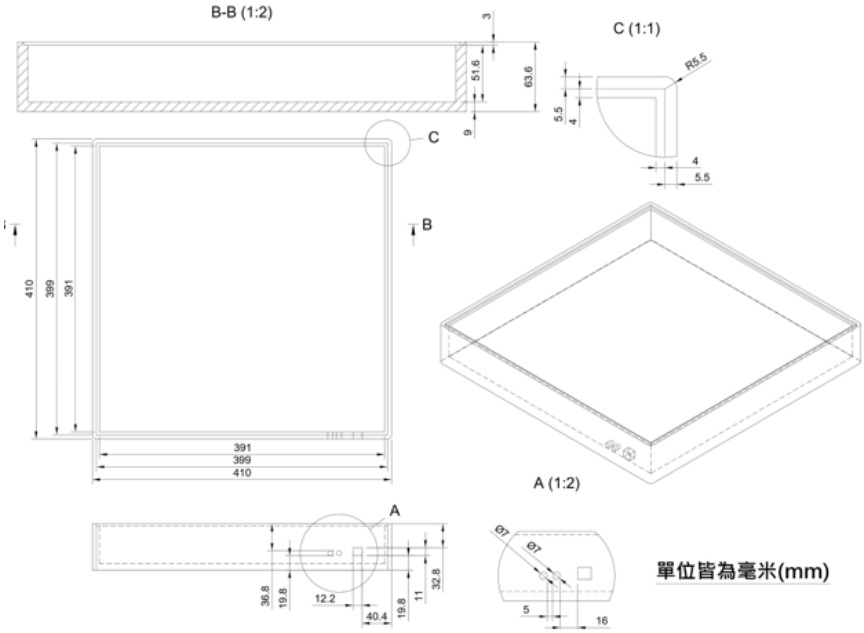
圖十五：磁簧開關陣列（上）與開發板（下）（資料來源：研究者自行拍攝）

而該陣列將會被放在棋盤正下方（如上圖十五），開發板會利用上圖十五中的八條紅線以及八條白線來做棋盤的掃描，來持續追蹤棋盤上棋子的位置，接著會將 LED 燈條由下往上固定在棋盤以及磁簧開關陣列的下方（如下圖十六），用來點亮棋盤上的格子以達成提示落子位置的功能。



圖十六：磁簧開關陣列（下）與 LED 燈條陣列（上）（資料來源：研究者自行拍攝）

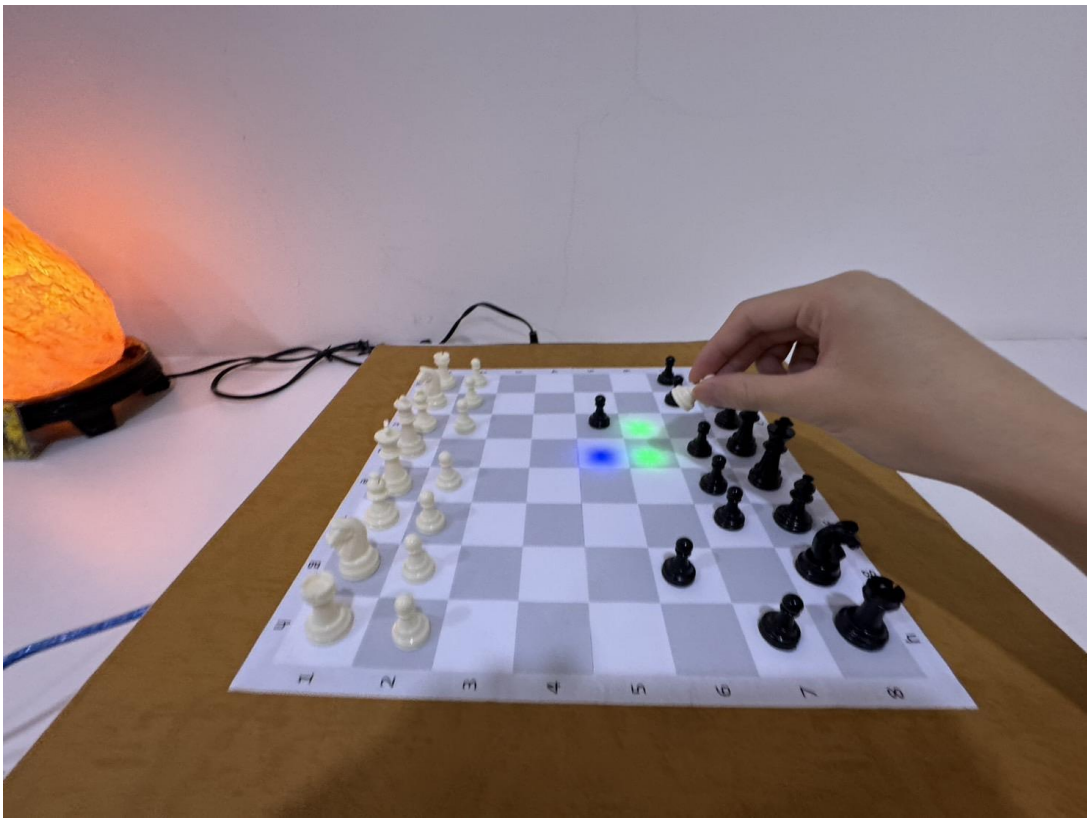
繪製棋盤底座設計圖（如圖十七），並利用 CNC 加工完成棋盤底座。



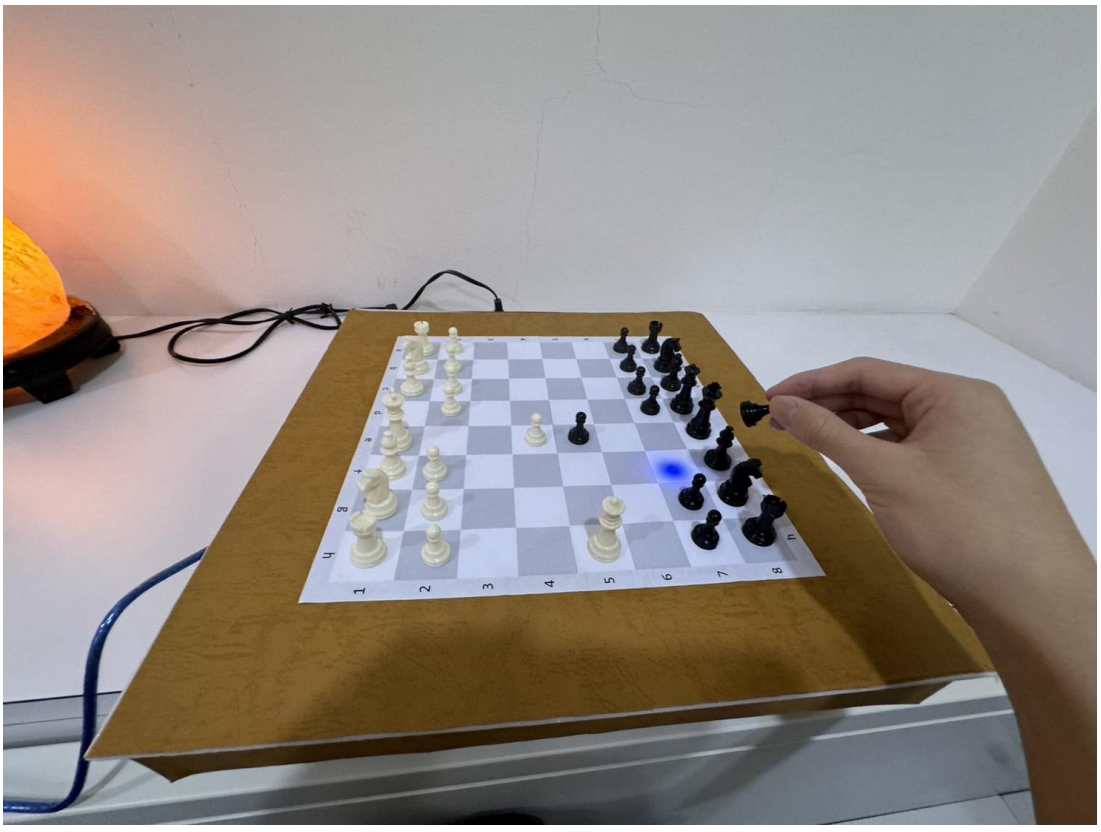

圖十七：棋盤底座設計圖（資料來源：研究者自行繪製）

最後將所有部件組裝起來：偵測旗子位置的磁簧開關陣列、擔任者個棋盤核心運算裝置的 Arduino UNO 開發板、用來給予使用者回饋的 LED 燈條陣列、提供給棋盤電力的行動電源以及可以把以上所有東西都包起來的棋盤外殼，組裝完的成品（如下圖十八至二十）。

圖十八：智慧棋盤組裝完成品（資料來源：研究者自行拍攝）

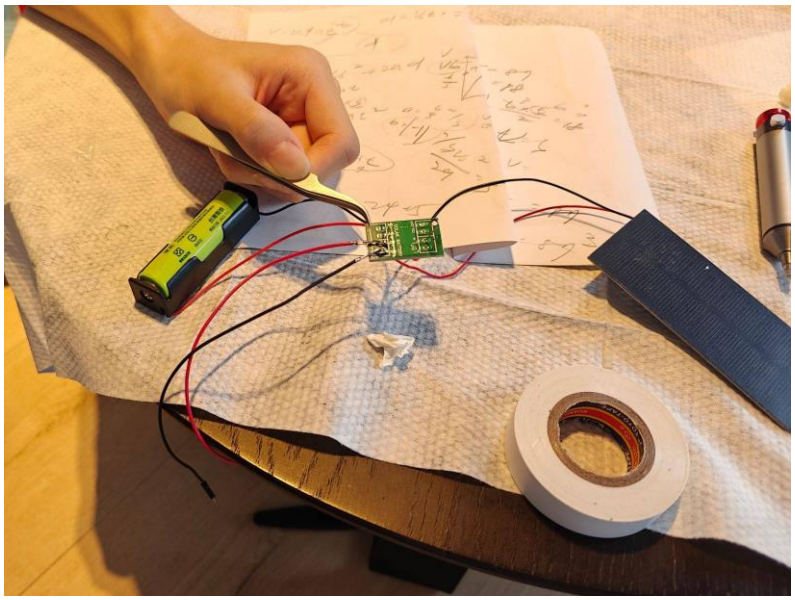


圖十九：智慧棋盤組裝完成品（圖中為小兵的合法棋步，包含過路兵）
（資料來源：研究者自行拍攝）

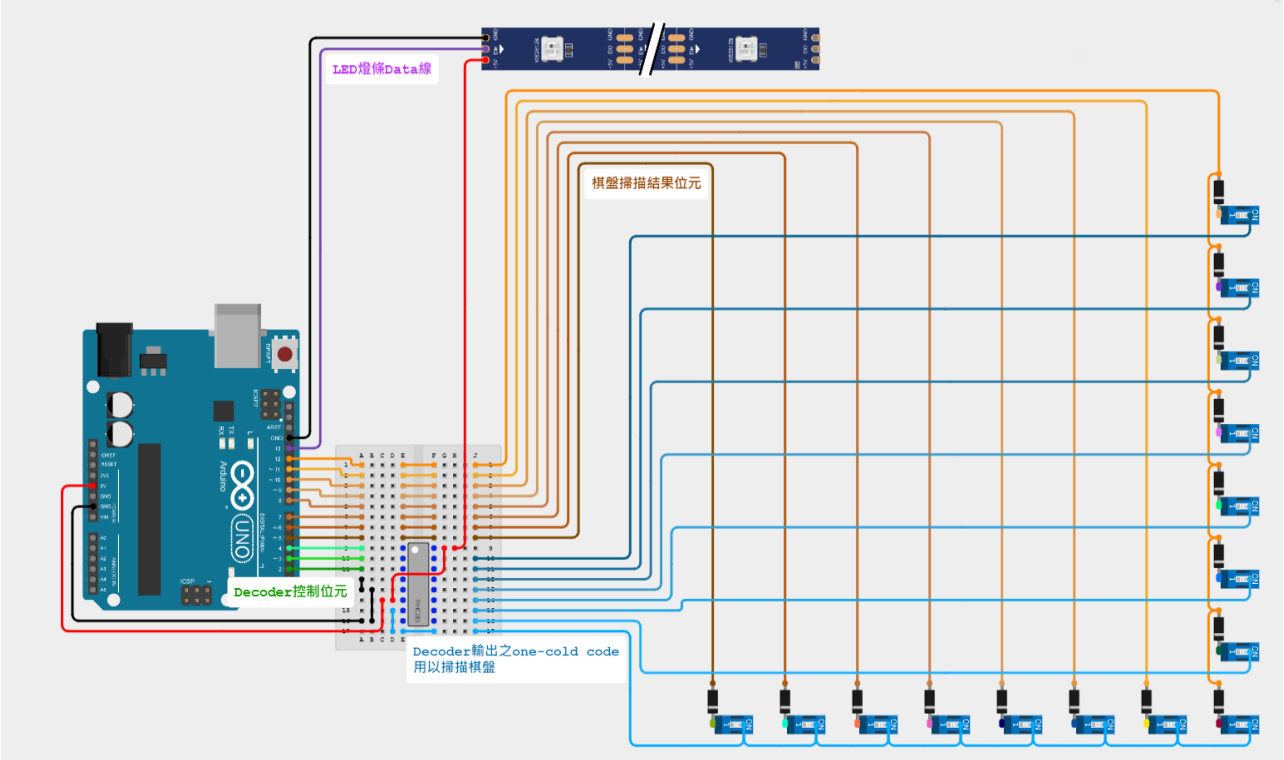


圖二十：智慧棋盤組裝完成品（圖中小兵無法往前，因為白方皇后正在攻擊國王）
（資料來源：研究者自行拍攝）

並且我們目前還還在棋盤中加入了太陽板（如下圖二十一），希望未來可以透過太陽能板進行開發板的供電。



圖二十一：太陽能板（右），充電電池（左）（資料來源：研究者自行拍攝）




圖二十二：智慧棋盤線路設計圖（同圖十四）（資料來源：研究者自行繪製）

接下來講解棋盤掃描原理，首先上圖二十二中「綠色系」的導線是用來當作 Decoder 的輸入，Arduino 會依序將此三個 Pin 設為 000~111（二進制），也就是十進制的 0~8，而 Decoder 輸出則會依照下圖二十三 Datasheet 中的真值表更新「藍色系」的 8 條導線，並且同一時間，只會有 1 條導線為低電位，剩餘 7 條皆為高電位（因為是使用 Arduino 內建的 Pull-up resistor，故用只需將用來掃描的一條導線設為低電位）。

於是我們就可以利用 8 條「橘色系」的導線來取得棋盤上特定一行的棋子擺放情況。最後再透過 Arduino 運算過後判斷盤面有無更新，並利用紫色的導線控制 RGB LED 燈條給予玩家相對應的回饋。

**Table 7-1. Function Table 'HC138, 'HCT138**

| INPUTS | | | | | | OUTPUTS | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STROBE | | | ADDRESS | | | | | | | | | | |
| $G2$ | $\overline{G1}$ | $\overline{G0}$ | $A_2$ | $A_1$ | $A_0$ | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ | $Y_7$ |
| X | X | H | X | X | X | H | H | H | H | H | H | H | H |
| L | X | X | X | X | X | H | H | H | H | H | H | H | H |
| X | H | X | X | X | X | H | H | H | H | H | H | H | H |
| H | L | L | L | L | L | L | H | H | H | H | H | H | H |
| H | L | L | L | L | H | H | L | H | H | H | H | H | H |
| H | L | L | L | H | L | H | H | L | H | H | H | H | H |
| H | L | L | L | H | H | H | H | H | L | H | H | H | H |
| H | L | L | H | L | L | H | H | H | H | L | H | H | H |
| H | L | L | H | L | H | H | H | H | H | H | L | H | H |
| H | L | L | H | H | L | H | H | H | H | H | H | L | H |
| H | L | L | H | H | H | H | H | H | H | H | H | H | L |

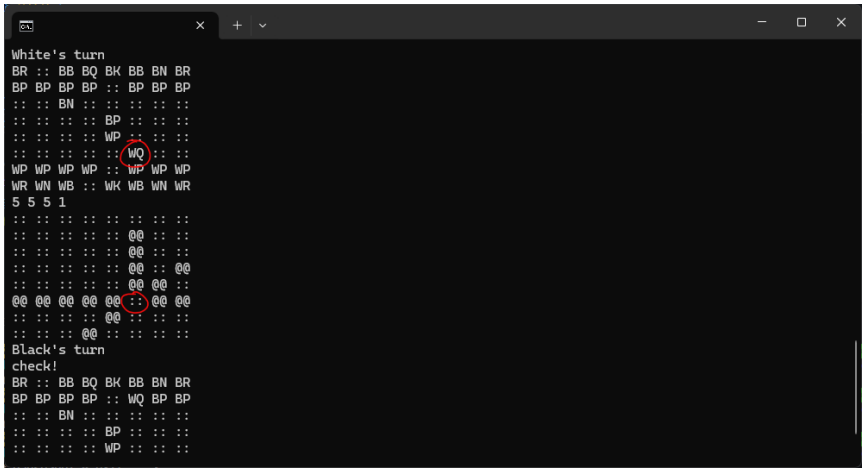
H = High Voltage Level, L = Low Voltage Level, X = Don't Care

圖二十三：74HC138 之真值表

（資料來源：https://www.ti.com/product/CD74HC138）

二、西洋棋基本規則

下圖二十四為顯示合法落子位置的示範，下圖二十四，視窗中@@符號代表著可以移動的範圍，也可以看到程式成功判定出將軍的情況。




圖二十四：合法落子位置顯示（資料來源：研究者自行製作）

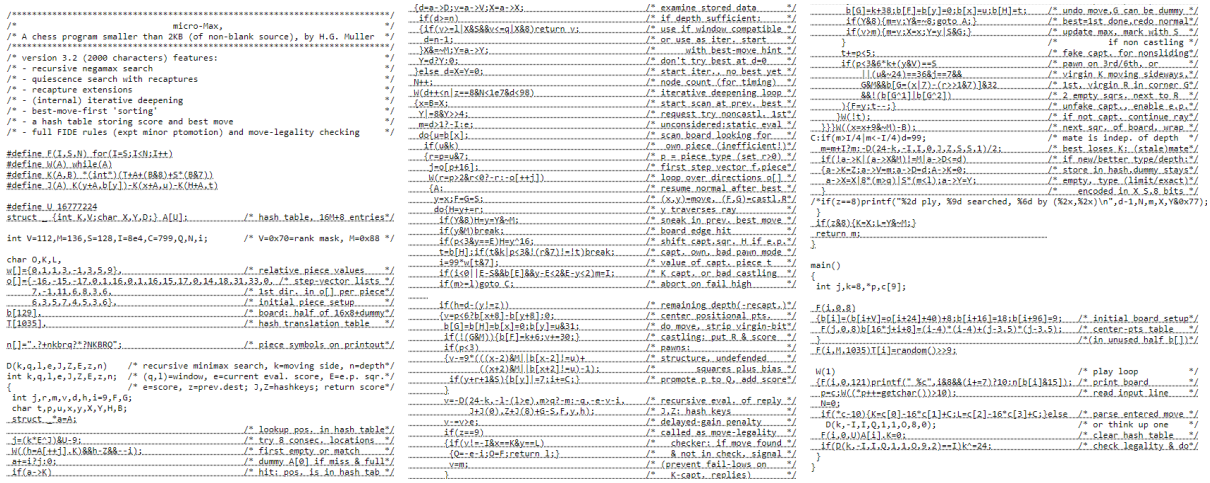
而判斷皇后之合法落子點之程式如下圖二十五，因為皇后可以走直線以及對角線，分別對應了城堡以及主教，故我們可以拿城堡以及主教的移動判定來使用，有效的減少內容及功能雷同的程式碼。

```
void getQueenMove(int _y, int _x, bool tar_moves[8][8], bool includeOwnPieces){
    getRookMove(_y , _x, tar_moves, includeOwnPieces);
    getBishopMove(_y, _x, tar_moves, includeOwnPieces);
}
```

圖二十五：皇后移動判定程式碼（資料來源：研究者自行製作）

三、Minimax 演算法

在一開始的版本中，加入的西洋棋 AI 則是使用 Minimax 演算法且程式碼只有 133 行，大小只有約 2KB 的 Micro-Max，如下圖二十六，之所以選用此 AI 是因為其輕量化的程式碼以及其縝密的程式邏輯，首先因為程式非常輕量化的關係，所以非常適合在運算速度以及記憶體大小都遠不及家用電腦的開發板上面運行，接著也因為該 AI 的程式碼之中的邏輯判斷、棋盤遍歷甚至是棋盤的界外判定等，大多都是利用非常高效率的位元運算來達成，是非常輕量化且高效率的西洋棋 AI，所以我們才會選擇使用它。



圖二十六：Micro-max 之原始碼

（資料來源：https://home.hccnet.nl/h.g.muller/max-src2.html）

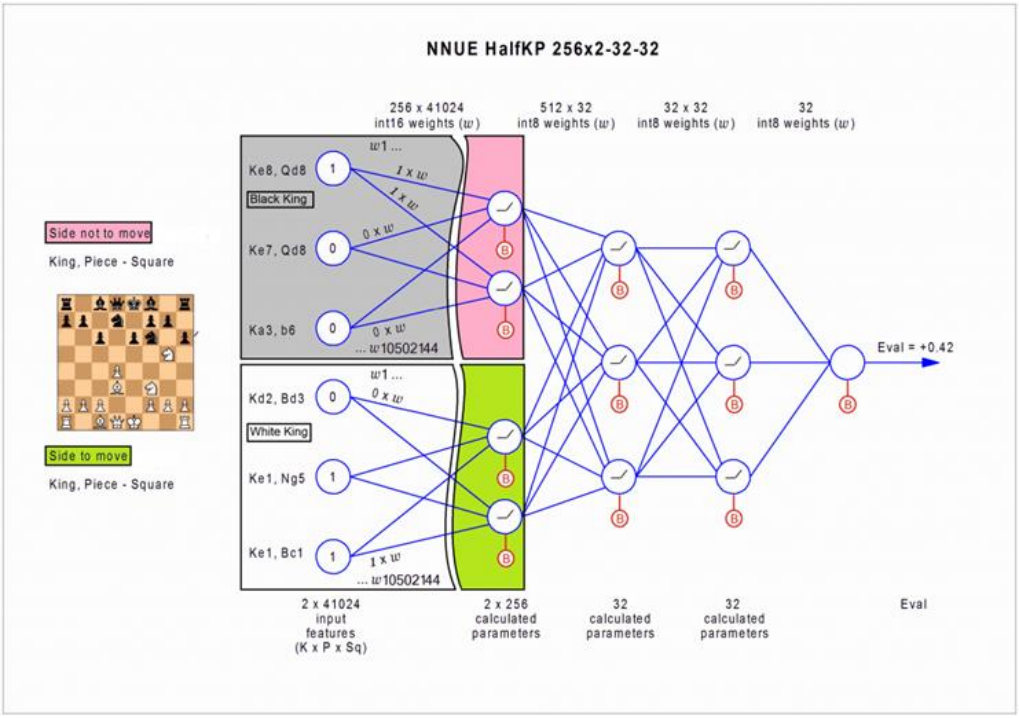
但也不是說這個 AI 就沒有缺點，像是其為了使程式精簡而使用的 Sliding pieces (城堡、主教以及皇后這幾個可以往特定方向滑動的棋子)棋步生成就是使用單純的 while 迴圈，但是若是使用更高效的查表方式來生成棋步，例如 Magic bitboard，又或是依據遊戲進程給予不同的位置審局分數加乘等，應該都會有助於提升該 AI 的棋力，卻因為需顧及程式碼的精簡而被犧牲，說不定是個未來我們可以試著修改的方向。

四、實作高效可更新神經網絡(NNUE)

單純使用 alpha beta 剪枝的西洋棋 AI 效果就已經不錯了，但是因為在開發板上運行，故遊戲樹的搜尋深度無法太深，故可能會有地平線效應(Horizon Effect)，所以本研究想要嘗試在葉節點中的審局函數中加入神經網路，來避免地平線效應的產生，但是在效能已非常吃緊的開發板上面加入神經網路，聽起來非常反直覺，畢竟神經網路太淺參數太少可能幫助不大，但參數多又會造成開發板無法負荷，於是我們就必須要介紹一個特別的網路架構：NUEE(ЭНИИ, Efficiently Updatable Neural Network)高效率可更新神經網路，此網路架構擁有更新非常高效率的特性等，是有可能提升開發板上西洋棋 AI 棋力的候選人之一。

在決定使用 NNUE 之前，我們曾嘗試使用較大型的神經網路（隱藏層約為 1000，共兩層）來嘗試製作西洋棋 AI，訓練次模型來進行審局評分，在生成出所有可行的棋步後，使用此模型來評估分數，並選擇分數最高的棋步，但是效果不彰，先撇除模型大小以及運算效能不說，棋力部分，只要是稍微有下過西洋棋的人都可以打的過它，若是

使用 alpha beta 剪枝的演算法棋力也比它高，接著回到模型大小以及效能，如果要將兩層有著 1000 個神經元的網路放進開發版中做推論，基本上是不太可能的事情，並且若是單純只搜尋遊戲樹的第一層並審局，會太高估模型的泛化能力，故我們想要嘗試尋找可高效推論並且可以應用在現有的 alpha beta 剪枝搜索上的神經網路，在查閱了網路上西洋棋 AI 的實作方法後，決定使用 NNUE（如下圖二十七），因為其高速的推論速度，以及其應用在 alpha beta 剪枝搜索上曾被證實具有一定的棋力(知名 Chess Engine 就使用了 NNUE 並且多度超越自身)，考慮到開發板的效能，此種實作方式將會有最大的機率能被應用在我們的西洋棋盤上。




圖二十七：NNUE 架構示意圖

（資料來源：https://www.chessprogramming.org/Stockfish_NNUE）

在正式介紹 NNUE 之前，以下為 NNUE 需遵循的三點原則：

（一）神經網路需要有相對少量的非零輸入（輸入矩陣需為稀疏矩陣）。

（二）兩次連續的神經網路推論之間，輸入的變動需盡量最小。

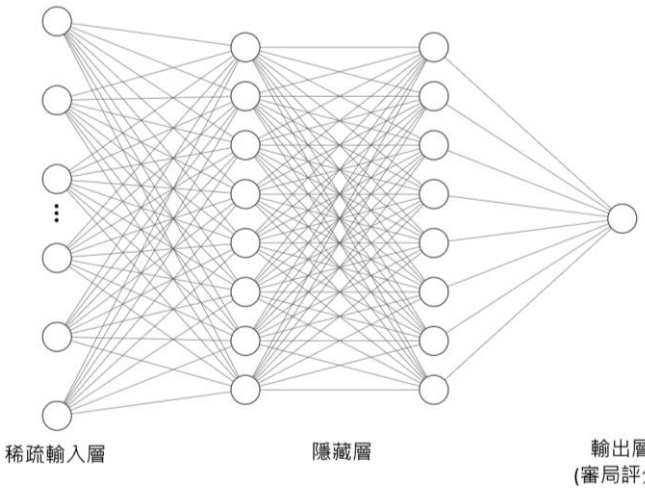
（三）因網路將會在整數域進行推論，故神經網路的架構必須要足夠簡單。

第一點是因為少量的非零輸入能夠讓網路推論的速度增加非常多，通常非零輸入會小於整體輸入的 1%，這是能夠讓 NNUE 推論速度加快的主要原因。

第二點則是利用在搜尋遊戲樹時，前一個評估的局面與下一個通常相差甚小，利用這點，並紀錄兩者相差的位置，就可以在下次評估時只計算兩個局面之間的差值，而不需要更新到整個網路，進而增加網路整體的效能。

第三點則是為了使用量化來達到最佳計算性能，所以若是模型複雜，就會讓推論速度變慢，且量化誤差逐漸累積。

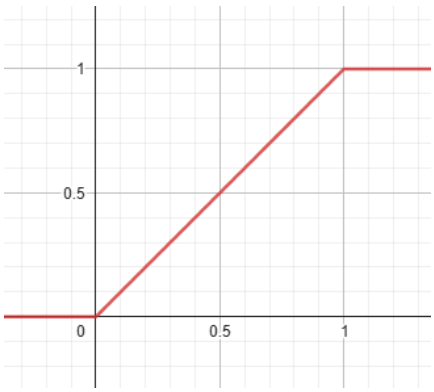
接下來我們將會一步一步介紹 NNUE 是如何做到高效能推論的，首先我們先從其線性層(Linear layer)開始說起，其為神經網路中最基礎的架構，他的運算可以使用矩陣運算來表示$y = Wx + b$其中 y 為輸出神經元的矩陣，x 為輸入神經元的矩陣，W 則是所有連結 x 到 y 層 weights 的矩陣，b 則是神經元的 bias，而 NNUE 特別的地方就在於，其網路第一層，也就是 input layer 中的神經元會是稀疏矩陣，也就是其中非常多輸入神經元的值皆為 0，只有極少數的神經元的值會不為 0，這是一個非常重要的特性，也是其中一項可以讓網路快速推論的原因之一，在繼續介紹之前，我們可以先看一下一個 NNUE 的架構示意圖（下圖二十八）。




圖二十八：NNUE 簡易架構圖（資料來源：研究者自行繪製）

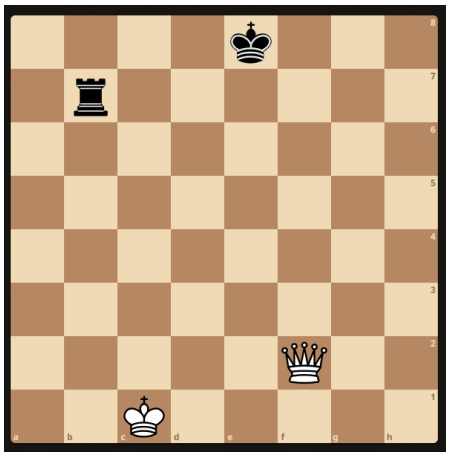
上圖中，輸入層的神經元數量通常會遠大於隱藏層，並且會將棋局的局面透過 feature set 對應成神經網路的輸入，而輸出層則是會輸出網路推論出的局面分數。

接著介紹網路中使用到的激勵函數(Activation function)，在神經網路中為了使其有非線性函數的特性，通常會加入激勵函數，而在 NNUE 中，則是使用 Clipped ReLU，其與一般 ReLU 的差別為，Clipped ReLU 被限制在 0~1 之間（如下圖二十九）。




圖二十九：Clipped ReLU 函數圖形（資料來源：研究者自行繪製）

之所以使用 Clipped ReLU 是因為之後的神經網路量化方式會需要把神經元的輸出值範圍限制住，若是使用傳統 ReLU 則會讓其輸出值值域太大。



圖三十：盤面範例（資料來源：研究者自行繪製）

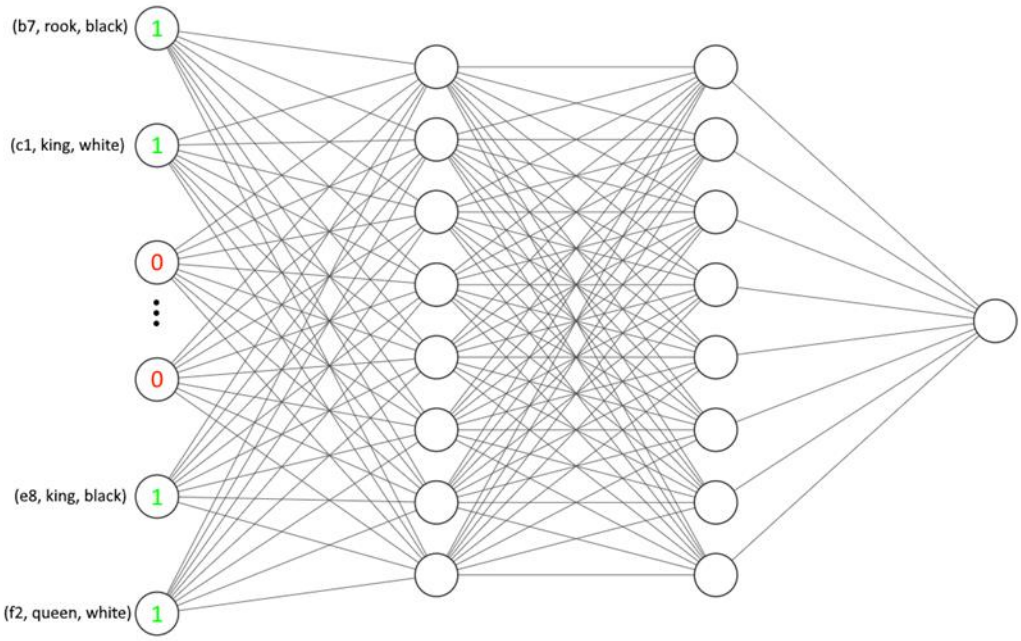
接著是神經網路輸入的格式，在這裡稱作 Feature set，通常 NNUE 的輸入層神經元數量會非常多，對推論的準確度有著舉足輕重的貢獻，好的 Feature set 也可以使網路的輸入更加稀疏，而更加符合前文提及的 NNUE 原則。

（棋子位置， 棋子類型， 棋子顏色）

來代表所有的神經元輸入，故此 Feature set 會有$64 \times 6 \times 2 = 768$個輸入神經元，若是用上圖三十的棋局來舉例，則此時神經網路的輸入就會呈現下圖三十一的狀態，除了

(b7, rook, black), (c1, king, white), (e8, king, black), (f2, queen, white)

四個神經元為 1 以外其餘 764 個皆為 0。

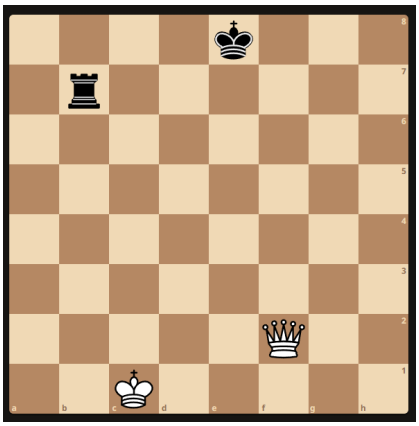



圖三十一：Feature set 於網路輸入層示意圖（資料來源：研究者自行繪製）

但是上述的 Feature set 只是用來舉利用的，實際上 StockFish 使用的 Feature set 是一個稱作 HalfKP 的 Feature set，他的規則則是：

（國王位置，棋子位置，棋子類型，黑方白方）

第一項代表的是我方國王的位置，第二項則是我方棋子的位置，第三項則是（前一項）不包含國王的棋子類型，最後則是黑方白方，這種 Feature set 的大小相對前文所舉出的例子來講多了很多，總共會有$64 \times 64 \times 5 \times 2 = 40960$個輸入神經元，我們再用前文提過的盤面來舉例：

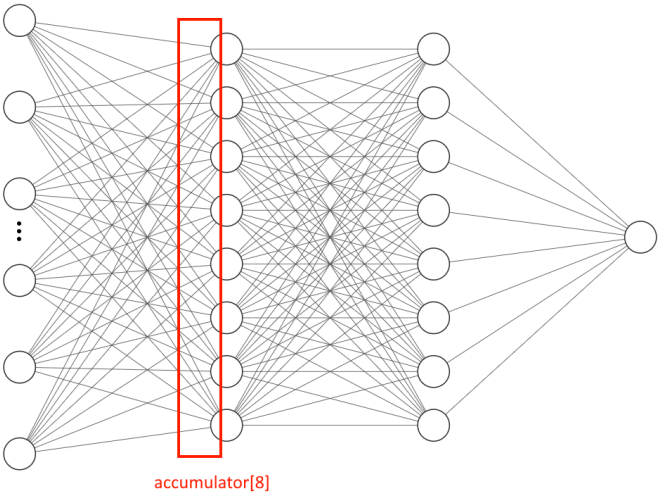


圖三十二：盤面示意圖（同圖二十七）（資料來源：研究者自行繪製）

在上圖三十二的盤面中，輸入的 HalfKP feature set 就會有：

(c1, f2, queen, white), (e8, b7, rook, black)

兩個神經元為 1，其餘皆為 0，由此可見使用這套 Feature set 可以更有效的達到 NNUE 的第一項原則，並藉由更加稀疏的輸入來增加網路的效能，且因此 Feature set 在第一項加入了國王位置的關係，所以訓練出來的模型會更加注意國王的位置，已達到更好的審局效果。




圖三十三：Accumulator 示意圖（資料來源：研究者自行繪製）

網路的推論，也就是正向傳遞，則會需要使用到追蹤差值的優化方式，所以在細部的實作程式中，我們會使用 Accumulator 來持續追蹤第二層中所有加權總和，如上圖三十三，對於第二層中的任意神經元$N_i^{Hidden}$來說，因為 NNUE 的特性使得網路的輸入只會有 0 和 1，故可以推得：

對於所有從 0 變為 1 的輸入神經元$N_j^{In}$
將其 weight[ i ][ j ] 值加入 accumulator[ j ]

對於所有從 1 變為 0 的輸入神經元$N_j^{In}$

將其 weight[ i ][ j ] 值從 accumulator[ j ]扣除

並且因之後網路會被量化，推論會在整數域上進行，所以頻繁的對 accumulator 中的數字增減並不會像浮點數一樣會有精準度的問題。

整體而言 NNUE 其簡單的架構以及高速的推論速度讓他成為了我們棋盤 AI 的人選，接著我們將三種 AI 進行比較：

表二：複雜網路模型、NNUE、Minimax 比較表（資料來源：研究者自行製作）

| | 複雜網路模型 | NNUE | Minimax |
|---|---|---|---|
| 優點 | 實作相對簡單 | 無地平線效應<br>可以再開發板上運行 | 運算速度較快<br>可以再開發板上運行 |
| 缺點 | 速度極慢<br>棋力低<br>無法再開發板上運行 | 運算速度較慢 | 有地平線效應 |

# 肆、 結論與應用

本研究成功製作出使用磁簧開關偵測棋子位置，用棋盤底下的燈光提供給玩家回饋之智慧棋盤。主要功能如下，當玩家拿起一顆棋子時，棋盤將會使用棋盤下的燈光標示出該子的合法落子位置、若玩家將棋子下在非法位置時會予以提醒、棋盤能夠自動判定將軍(Check)以及輸贏(Check mate)並給予相對應的燈光回饋、在玩家進行多子（國王入堡）或其他較複雜（吃過路兵）的移動時，棋盤也會給予燈光提示，最後棋盤也加入了能夠讓玩家自行其對弈的 AI 機器人。

我們利用磁簧開關陣列，以二極體防止 Ghosting 效應，並利用 Arduino UNO，作為程式運算載體，設計並製作出智慧棋盤的硬體設備。此棋盤於設計初以輔導新手玩家落子為目標，未來希望能加入棋鐘與不同計時模式選擇，供入門棋手對弈時使用，以拓展本智慧棋盤之適用範圍。

我們也在棋盤中加入了太陽能板以及充電電池來更大的增加棋盤的便利性，讓棋盤在照到太陽時可以充電，並通過電池供電。雖然目前電池的容量並不大，但未來我們計劃加入更多顆電池或是使用單顆容量較大的電池來增加續航力，也會調低 LED 燈條的閃爍頻率，進而在肉眼看不出差別的前提下，達到更加省電的效果。

而在加入 AI 的版本中，為達到節省運算效能的目的，我們選用輕量化的西洋棋 AI，並成功達成原設定目標，該 AI 相較於市面上眾多應用程式棋力皆位於中上階段，並具備運算速度快之特點，平均於玩家落子後十秒內落子，省去玩家等待時間的同時具備一定的挑戰性與訓練棋力的目的。若能以運算效能較高的載體，可以嘗試以不同演算法製作，並提供玩家不同難度選擇的功能。

NNUE 的部分則是希望在未來可以將原本的 Arduino Uno 使用 ESP32 S3 替換掉，原因是因為它有較高的微控制器時脈，可以使我們的搜索效率增加，除此之外，它還支援 SIMD(Single instruction, multiple data)，所以就可以完全受惠於 NNUE 中量化所帶來的速度，再者，其對藍牙以及 WiFi 的支援提供了日後若是要與電腦或智慧型手機連結的可能性。

未來希望能夠加入更多優化方式，例如 Bitboard、Move ordering、Iterative deepening、Transposition table 和 Quiescence search 等，以及改善程式碼中的資料結構和程式碼整體架構等，以增加每秒可以拓展的步數，畢竟若是要能夠在開發板上運行，就必須要納入非常多的優化方法，使得其在具備足夠棋力的同時，在搜索速度上能更迅速。

# 伍、 參考文獻


[1] Chess Programming Wiki.(2024, July 7) Stockfish NNUE

[2] Circuit Geeks.(2022, March 2) How to Control WS2812B Addressable RGB LEDs using Arduino

[3] Daniel Eubanks.(2019, July 5) Connect Four Minimax Algorithm

[4] Eric Rosado.(2018, Jan 2) Alpha Beta Pruning In Minimax

[5] H.G. Muller.(2006, February 7) Micro-Max, a 133-line Chess Source

[6] James Lewis.(2017, December 15) Arduino Keyboard Matrix Code and Hardware Tutorial

[7] Stuart Russell, Peter Norvig.(2021) Artificial Intelligence: A Modern Approach, 4th ed. Hoboken : Pearson.

[8] Texas Instruments(2021, 17 Nov) CD74HC138 High Speed CMOS Logic 3-to-8 Line Decoder Demultiplexer Inverting and Non-Inverting

[9] Yu Nasu.(2018, April 28) Efficiently Updatable Neural-Network-based Evaluation Functions for Computer Shogi

# 【評語】190019

1. 此作品為結合軟硬體的智慧型西洋棋棋盤，具有創意。

2. 惟對於此西洋棋棋盤的使用者族群(例如具初階或高階棋力者)的設定部分可以進一步確認。

3. 建議此智慧型西洋棋棋盤可以透過連網功能的發展，讓系統推薦棋步的棋力可以動態調整。