微算機實驗 期中專題

組別：第五組

組員：410985045 胡凱翔 410985034 陳麒升

一、題目：黑白棋

利用：雙點矩陣、4*4 keyboard、蜂鳴器

需要 include <pitch.h>、<Keyboard.h>

雙點矩陣 pin 腳位：

Dot-Matrix(P26)	Mega2560	Dot-Matrix(P27)	Mega2560	Dot-Matrix(P28)	Mega2560
C1	53	G1	37	R1	36
C2	51	G2	35	R2	34
C3	49	G3	33	R3	32
C4	47	G4	31	R4	30
C5	45	G5	29	R5	28
C6	43	G6	27	R6	26
C7	41	G7	25	R7	24
C8	39	G8	23	R8	22

圖一

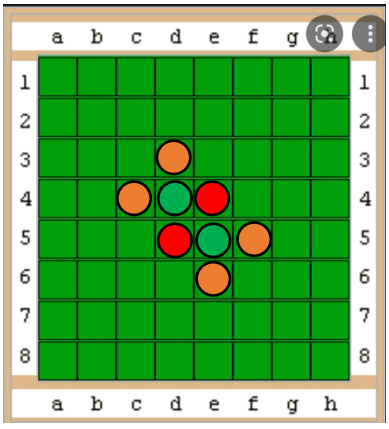
4*4 Keyboard 腳位：

4*4鍵盤(P40)	Mega2560
ROW1	4
ROW2	5
ROW3	6
ROW4	7
COL1	8
COL2	9
COL3	10
COL4	11
EINT	空接

蜂鳴器腳位：

BUZZER(BZ1)	Mega2560
BUZZER	3

二、原理：



創一個物件 Reversi（黑白棋，本專題使用紅、綠棋），橘色棋為提示該玩家可以下棋的位置，其他未顯示位置的地方不能下，一開始的狀態會如右圖，紅方先下。

一局最多只會有 60 game（64-4（4 個一開始放在棋盤上的棋）），所以利用一個迴圈來結束一局，每一回合一開始呼叫 find_placeable_grid 傳入玩家代碼（RED = 1,GRN = 2）利用迴圈檢查棋盤上的空位置，往 8 個方向做搜尋，如果在搜尋的路徑上連續出現數個對手的棋子，再遇到自己的棋子，代表這個位置可以作為下子的位置，並將可以下子的位置顯示橘色（如右圖），若至少有一個位置可以下子就回傳 true，若無位置下子就回傳 false，此時玩家須放棄本回合中下子的機會，直接進入對手的回合。

接著把這一回合的玩家代碼（RED = 1,GRN = 2）傳入 playdisk，玩家可以選擇投降（按兩次左下角按鈕），遊戲就會結束。玩家要輸入這局要下的棋子位置（第一個輸入為 row，第二個為 col），只能輸入 0~7（利用 is_in_board 判斷兩輸入位置在棋盤內）且位置為橘色提示棋上，如果不符合以上條件則會發出聲音（alertsound），需重新輸入。如果符合上述條件則該位置開始閃爍，其他橘色提示位置會暫時不亮，按下一次確認鍵（右下角）才算下棋完成，如果想修改則按其他鍵重新輸入。下棋完成後呼叫 changedisk，將被我方棋子夾住的敵方棋子翻轉為我方棋子，接著再利用 resetboard 把提示的橘色棋改為 EPT，取消提示狀態。

如果遊戲下完（雙方都無法再下棋為止），則跳出此迴圈，會利用 countdisk 來計算綠方及紅方的分數，雙方確認完自己的棋子後，按下一次確認鍵(右下角)，接著利用 printL_W 印出 W 字樣，顯示的顏色為該方獲勝，如果雙方旗子一樣則顯示橘色燈，遊戲結束。

三、程式碼：

```
#include <Keypad.h>
#include "pitch.h"
int col[8]={39,41,43,45,47,49,51,53};
int red[8]={36,34,32,30,28,26,24,22};
int grn[8]={37,35,33,31,29,27,25,23};
#define  pin3  3///buzzer
const int LEN = 8;
const byte ROWS = 4;//four rows
const byte COLS = 4;//four columns
enum {EPT = 0, RED, GRN, MIX};
enum {NOHINT = 0,HINT = 1};
//                          u  ur   r  dr   d  dl   l  ul
const int displaceX[8] = { 0,  1,  1,  1,  0, -1, -1, -1};
const int displaceY[8] = {-1, -1,  0,  1,  1,  1,  0, -1};
char hexaKeys[ROWS][COLS] =
                          {  {'0','1','2','3'},
                             {'4','5','6','7'},
                             {'R','R','R','R'},
                             {'8','R','R','9'} };

byte rowPins[ROWS] = {4,5,6,7};
byte colPins[COLS] = {8,9,10,11};
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, 4, 4);
int letterW[8][8] =
 {1,1,1,1,1,1,1,1,
  0,1,1,1,1,1,1,0,
  0,1,1,1,1,1,1,0,
  0,1,1,1,1,1,1,0,
  0,0,1,0,0,1,0,0,
  1,0,1,0,0,1,0,1,
  1,0,0,0,0,0,0,1,
  1,1,0,1,1,0,1,1};
class Reversi {
  public:

  int board[8][8] =
    {
    EPT,EPT,EPT,EPT,EPT,EPT,EPT,EPT,
    EPT,EPT,EPT,EPT,EPT,EPT,EPT,EPT,
    EPT,EPT,EPT,EPT,EPT,EPT,EPT,EPT,
```

```
    EPT,EPT,EPT,GRN,RED,EPT,EPT,EPT,
    EPT,EPT,EPT,RED,GRN,EPT,EPT,EPT,
    EPT,EPT,EPT,EPT,EPT,EPT,EPT,EPT,
    EPT,EPT,EPT,EPT,EPT,EPT,EPT,EPT,
    EPT,EPT,EPT,EPT,EPT,EPT,EPT,EPT,
    };

    void show_board(int state,int place = -1);
    bool find_placeable_grid(int player);
    bool is_in_board(int x, int y);
    void place_disk(int player);
    void alertsound();
    void printL_W(int player);
    int countdisk();
    void changedisk(int row,int col,int player);
    bool surrender = false;
    void resetboard();
    void check_board();
};
void Reversi::resetboard(){
  for(int r = 0 ; r < 8 ; ++r){
    for(int c = 0 ; c < 8 ; ++c){
      if(board[r][c] == MIX) board[r][c] = EPT;
    }
  }
}
void Reversi::show_board(int state,int place = -1) {
    for (int c = 0 ; c < LEN; c++) {
        digitalWrite(col[c],LOW);
        for (int r = 0; r < LEN; r++) {
            if(state == NOHINT){        // print normal grid
                switch (board[r][c]) {
                    case RED: digitalWrite(red[r], LOW); break;
                    case GRN: digitalWrite(grn[r], LOW); break;
                }
            }else if(state == HINT){  // print placeable grid
                switch (board[r][c]) {
                    case RED: digitalWrite(red[r], LOW); break;
                    case GRN: digitalWrite(grn[r], LOW); break;
                    case MIX: digitalWrite(red[r], LOW);
                              digitalWrite(grn[r], LOW);
                }
            }else{                  // flicker grid
                int row = place/8;
                int col = place%8;
                int i = state - 2;
                if(row == r && col == c){
                    if(i/4%2 == 0){ // flick effect
                        digitalWrite(red[r],LOW);
                        digitalWrite(grn[r],LOW);
                    }
                }else{
                    switch (board[r][c]) {
                        case RED: digitalWrite(red[r], LOW); break;
                        case GRN: digitalWrite(grn[r], LOW); break;
                    }
                }
            }
        }
        delay(1);
        for (int r = 0; r < LEN; r++) {
          digitalWrite(red[r], HIGH);
          digitalWrite(grn[r], HIGH);
        }
        digitalWrite(col[c],HIGH);

    }
}
/*
```

```
bool Reversi::find_placeable_grid(int player) {
  bool canplace = false;
  int opposite = 3 - player;
  for (int r = 0; r < LEN; r++) {
    for (int c = 0; c < LEN; c++) {
        if (board[r][c] == EPT) {
            for (int dir = 0; dir < 8; dir++) {
                bool findOppo = false;
                int curX = c;
                int curY = r;
                while (true) {
                  curX += displaceX[dir];
                  curY += displaceY[dir];
                  if (!is_in_board(curX,curY))
                    break;
                  if (board[curX][curY] == opposite)
                    findOppo = true;
                  else
                    break;
                }
                if (findOppo && board[curX][curY] == player) {
                    canplace = true;
                    board[r][c] = MIX;
                    break;
                }
            }

        }
    }
  }
  return canplace;
}
*/
bool Reversi::find_placeable_grid(int player) {
  bool canplace = false;
  int opposite = 3 - player;
  for (int r = 0; r < LEN; r++) {
    for (int c = 0; c < LEN; c++) {
        if (board[r][c] == EPT) {
            for (int dir = 0; dir < 8; dir++) {
                bool findSelf = false;
                bool findOppo = false;
                int curr = r;
                int curc = c;
                while (is_in_board(curr+=displaceX[dir],curc+=displaceY[dir])) {
                  if(board[curr][curc] == opposite){
                    findOppo = true;
                  }else if(board[curr][curc] == player){
                    findSelf = true;
                    break;
                  }else{
                    break;
                  }
                }
                if (findSelf && findOppo) {
                    canplace = true;
                    board[r][c] = MIX;
                    break;
                }
            }

        }
    }
  }
  return canplace;
}

bool Reversi::is_in_board(int x, int y) {
  return (0 <= x && x < LEN && 0 <= y && y < LEN);
```

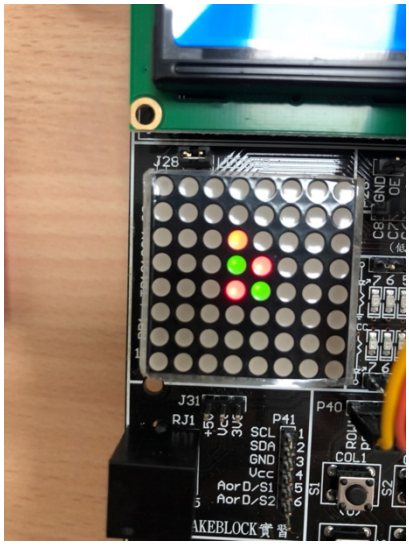
```
}
void Reversi::alertsound(){
    for (int i = 0 ; i < 4 ; i++) {
      tone(pin3, NOTE_C6, 200);
      delay(250);
    }
}
int Reversi::countdisk(){
    int red = 0;
    int grn = 0;
    for(int r = 0 ; r < 8 ; ++r){
        for(int c = 0 ; c < 8 ; ++c){
            if(board[r][c] == RED) ++red;
            if(board[r][c] == GRN) ++grn;
        }
    }
    Serial.println(red);
    Serial.println(grn);
    if(red > grn){
        return RED;
    }else if(grn > red){
        return GRN;
    }else{
       return MIX;
    }
}
void Reversi::printL_W(int player){
    for(int i = 0 ; i < 1000 ; ++i){
        for (int c = 0 ; c < LEN; c++) {
            digitalWrite(col[c],LOW);
            for (int r = 0; r < LEN; r++) {
                if(!letterW[r][c]){
                    switch (player){
                        case RED: digitalWrite(red[r], LOW); break;
                        case GRN: digitalWrite(grn[r], LOW); break;
                        case MIX: digitalWrite(red[r], LOW);
                                  digitalWrite(grn[r], LOW);
                    }
                }
            }
            delay(1);
            for (int r = 0; r < LEN; r++) {
              digitalWrite(red[r], HIGH);
              digitalWrite(grn[r], HIGH);
            }
            digitalWrite(col[c],HIGH);
        }
    }
}
void Reversi::changedisk(int row,int col,int player){
    int opposite = 3 - player;
    board[row][col] = player;
    for(int dir = 0 ; dir < 8 ; ++dir){
        bool findsame = false;
        int count = 0;
        int r = row;
        int c = col;
        while(is_in_board(r+=displaceX[dir],c+=displaceY[dir])){
            if(board[r][c] == opposite) ++count;  // continue
            else if(board[r][c] == player){     // stop to find
                findsame = true;
                break;
            }else break;                        // mix or empt
        }
        if(findsame){
            for(int i = 0 ; i < count ; ++i) board[r-=displaceX[dir]][c-
=displaceY[dir]] = player;
        }
    }
```

```
}
void Reversi::place_disk(int player) {
    char place[3] = {0};
    bool re_enter = false;
    int index = 0;
    int count = 0;
    while(index < 3){
        char input = customKeypad.getKey();
        if (input){
            re_enter = false;
            if(input == 'R'){
                index = 0;
                continue;
            }
            if(index == 2){
                if(input == '9') break;
                else re_enter = true;
            }
            if(index == 1){
                place[1] = input -'0';
                if(is_in_board(place[0],place[1])){
                    if(board[place[0]][place[1]] != MIX) re_enter = true;
                }else if(place[0] == 8 && place[1] == 8){
                    alertsound();
                    surrender = true;
                    return;
                }else re_enter = true;
            }
            if(re_enter){
                alertsound();
                index = 0;
            }else{ // index is 0 or could place on disk
                place[index] = input - '0';
                ++index;
            }
        }
        if(index == 2){
            show_board(count%=10000+2,place[0]*8+place[1]);
        }else show_board(1);
        ++count;
    }
    changedisk(place[0],place[1],player);
}
void Reversi::check_board(){
    while(true){
        char input = customKeypad.getKey();
        if(input){
            if(input == '9') break;
        }
        show_board(0);
    }
}
void setup() {

  // put your setup code here, to run once:
    Serial.begin(9600);  // 設定 Serial Monitor 的傳輸鮑率
    for(int i=0;i<=7;i++)
    {
      pinMode(col[i],OUTPUT);
      digitalWrite(col[i],HIGH);
      pinMode(grn[i],OUTPUT);
      digitalWrite(grn[i],HIGH);
      pinMode(red[i],OUTPUT);
      digitalWrite(red[i],HIGH);
   }
}

void loop() {
    Reversi reversi;
```

```
    for(int game = 0 ; game < 60 ; ++game){
         bool canplace = reversi.find_placeable_grid(game%2+1);
         if(!canplace){
            continue;
         }
         reversi.place_disk(game%2+1);
         reversi.resetboard();
         if(reversi.surrender){
            reversi.printL_W(3-(game%2+1));
            break;
         }
    }
    if(!reversi.surrender){
      int winner = reversi.countdisk();
      reversi.check_board();
      reversi.printL_W(winner);
    }
}
```

四、實驗結果：

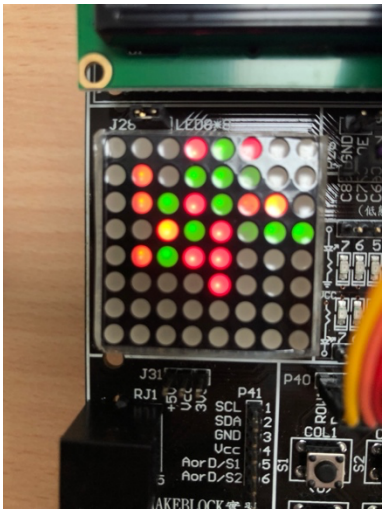
遊戲一開始的狀態



選定位置的狀態（橘燈其實是一閃一閃）

紅方獲勝顯示 W 字樣

遊戲過程
綠方回合時的狀態
橘燈為提示



遊戲過程
紅方回合時的狀態
橘燈為提示