2024 Final Project
軍艦棋(海戰棋) 實作

電機系 27 陳睿倬

112061105

專題簡介

軍艦棋(海戰棋):

雙方玩家將自己的軍艦布置在棋盤上，並去猜對方玩家軍艦的位置。

過程是由回合制進行的，由一方先猜另一方軍艦的位置。倘若猜對其中一個位置可以繼續連猜。反之，猜錯就換對方猜我方的軍艦位置。

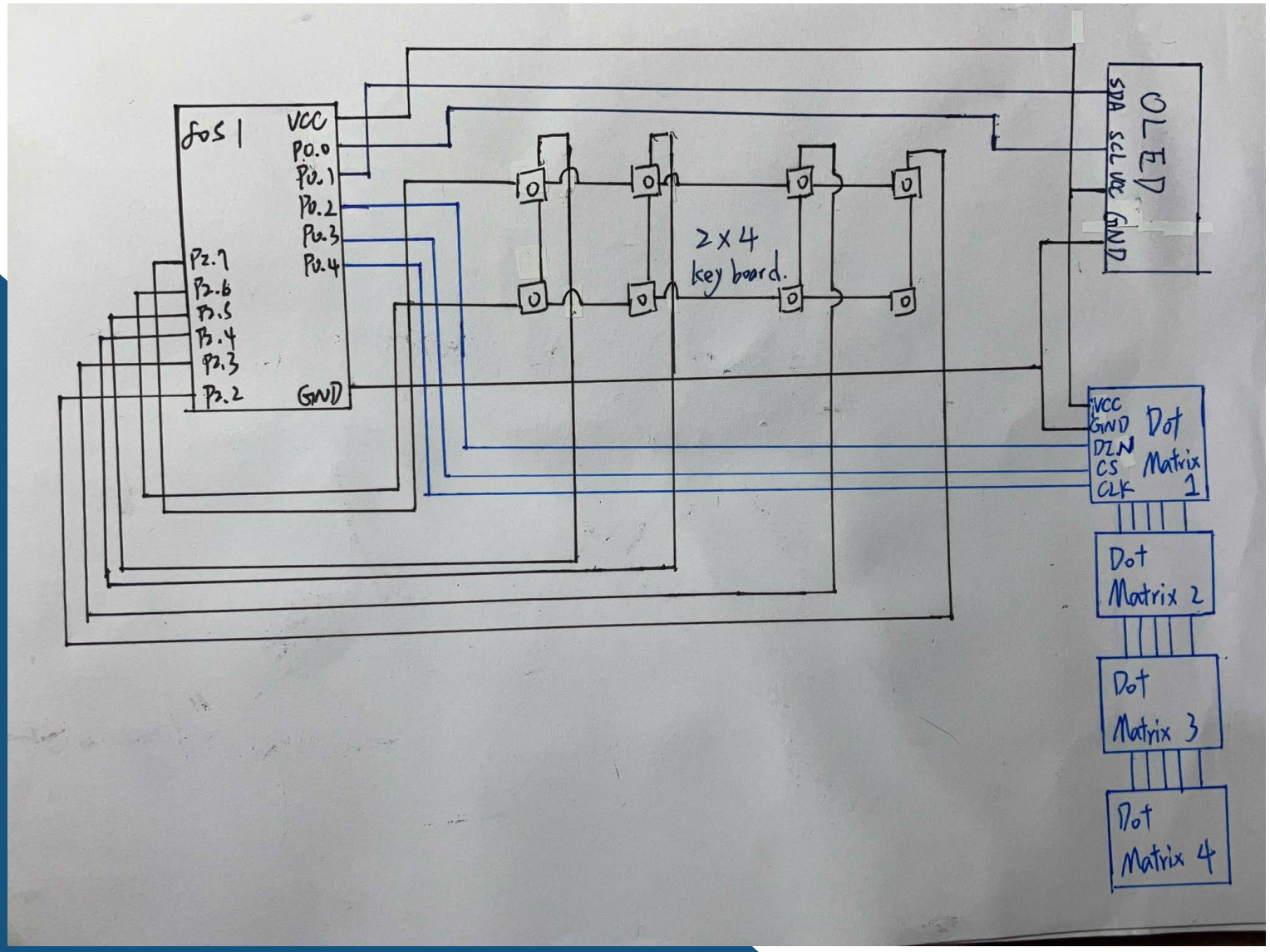
最先猜出對手全部軍艦的位置的人就贏了。

詳細遊玩流程請看Demo影片

程式皆是由本人完成，無參考其他資源。

海戰 (遊戲) wiki介紹

規格、佈線



MAX7219 點矩陣顯示器
(matrixnum 4)
DIN P0_2
CS P0_3
CLK P0_4

OLED
SCL P0_0
SDA P0_1

2*4 key_board
row1 out P2_7
row2 out P2_6
col1 in P2_5
col2 in P2_4
col3 in P2_3
col4 in P2_2

流程、設計

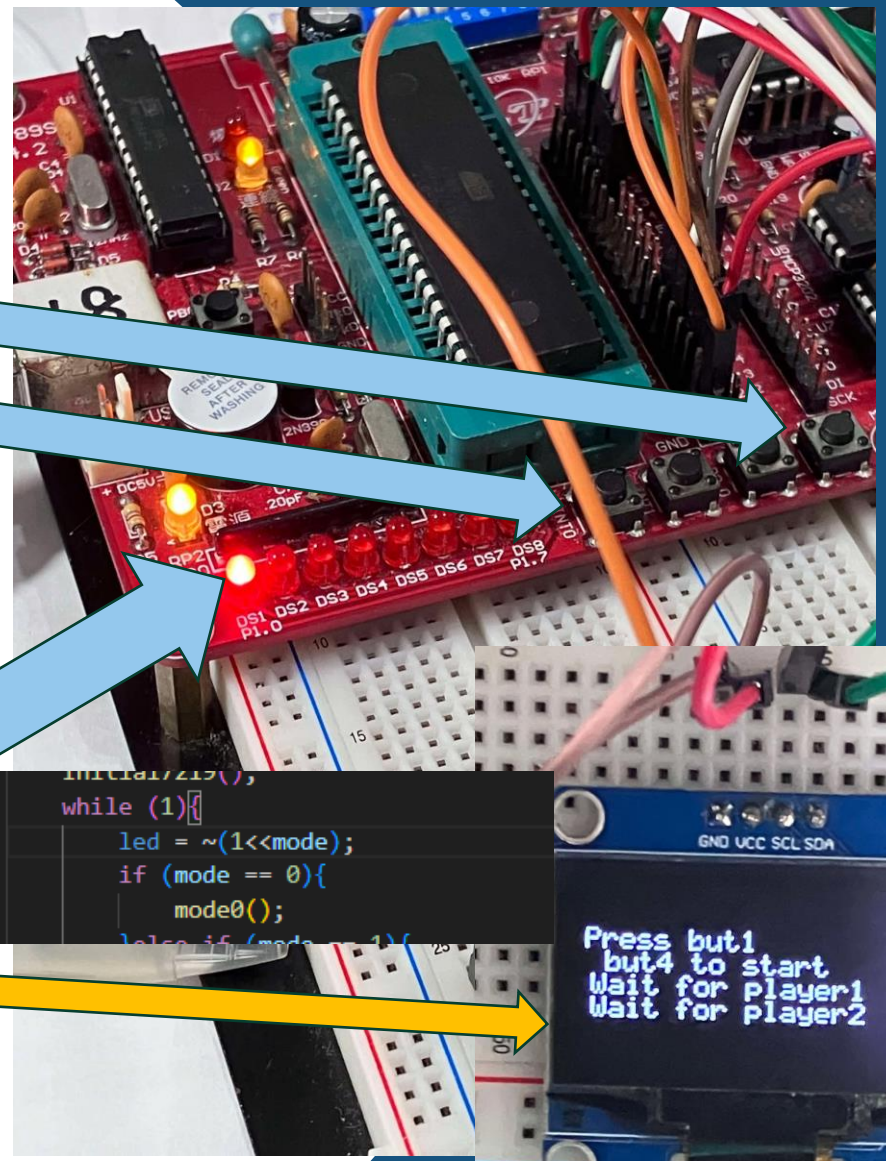
Mode0
雙方玩家準備階段(按下PB1, PB4開始)
Mode1
玩家1 設置軍艦位置(按PB1來確定軍艦擺放位置)
Mode2
玩家2設置軍艦位置(按PB4來確定軍艦擺放位置)
Mode3
玩家1猜玩家2的軍艦位置(按PB2來猜對方軍艦位置)
Mode4
玩家2猜玩家1的軍艦位置(按PB3來猜對方軍艦位置)
Mode5
玩家X 獲勝

流程、設計: mode0

```c
void mode0(void){
    if (but1 == 0){
        delay_ms(10);
        if (but1 == 0){
            player_bool[1] = 1;
        }
    }
    if (but4 == 0){
        delay_ms(10);
        if (but4 == 0){
            player_bool[2] = 1;
        }
    }
    OLED_SetCursor(2, 3);
    OLED_DisplayString("Press but1");
    OLED_SetCursor(3, 10);
    OLED_DisplayString("but4 to start");
    OLED_SetCursor(4, 5);
    if (player_bool[1] != 0){
        OLED_DisplayString("Player1 is ready");
    }else{
        OLED_DisplayString("Wait for player1");
    }
    OLED_SetCursor(5, 5);
    if (player_bool[2] != 0){
        OLED_DisplayString("Player2 is ready");
    }else{
        OLED_DisplayString("Wait for player2");
    }

    if (player_bool[1] != 0 && player_bool[2] != 0){
        mode = 1;
        player_bool[1] = 0;
        player_bool[2] = 0;
    }
}
```

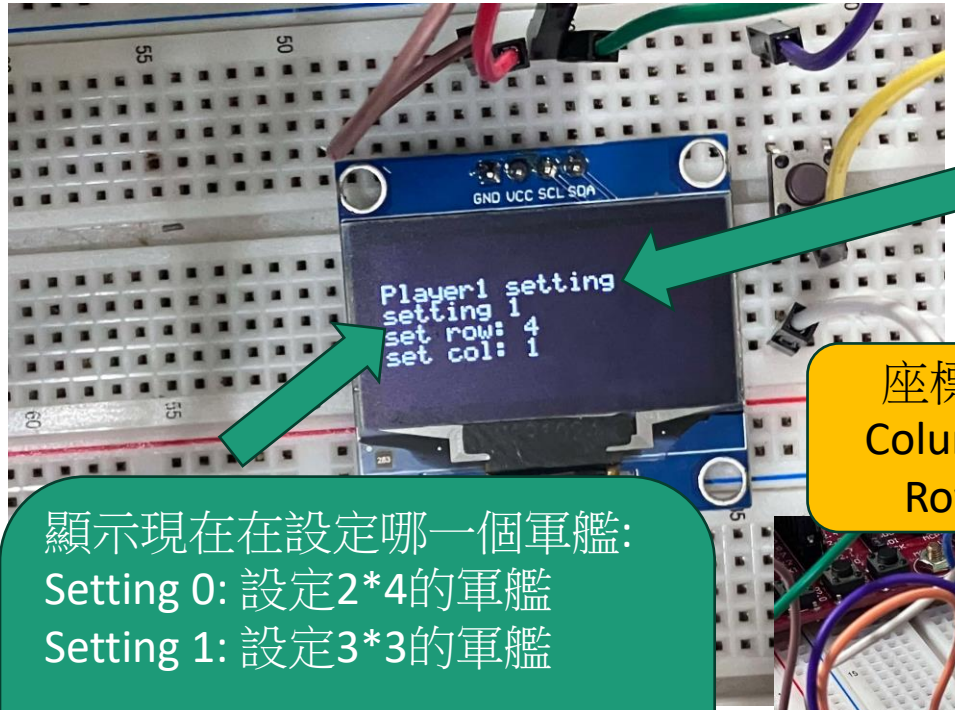
玩家1和玩家2分別按下PB1和PB4開始遊戲

左邊隔n燈沒亮就是代表mode n。
現在是最左邊亮所以代表mode0




```c
while (1){
    led = ~(1<<mode);
    if (mode == 0){
        mode0();
```

流程、設計: mode1

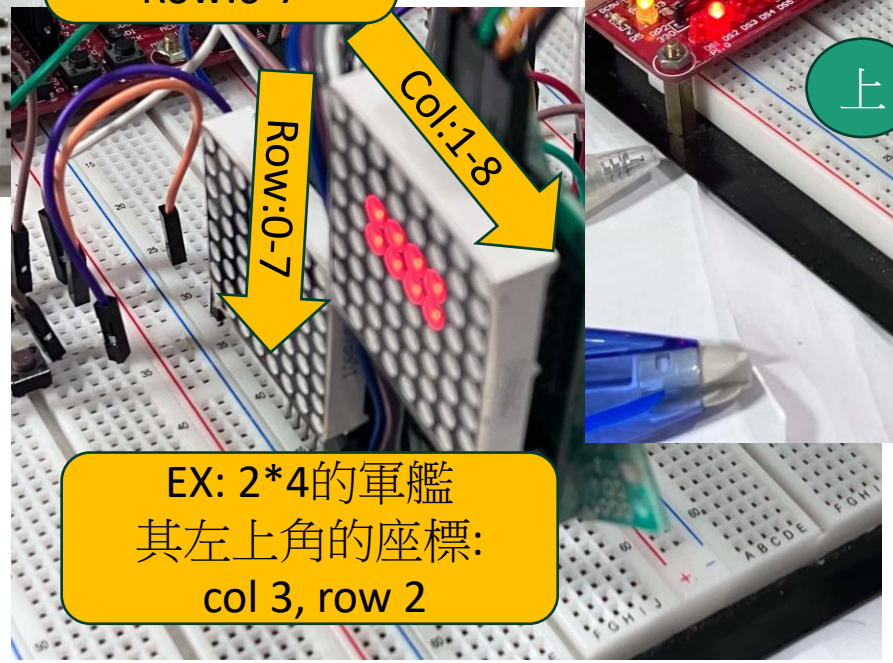


玩家1 的回合
放置軍艦

顯示現在在設定哪一個軍艦:
Setting 0: 設定2*4的軍艦
Setting 1: 設定3*3的軍艦

規定先設置軍艦0再設置軍艦1
下面兩行顯示現在控制的軍艦
其左上角的座標

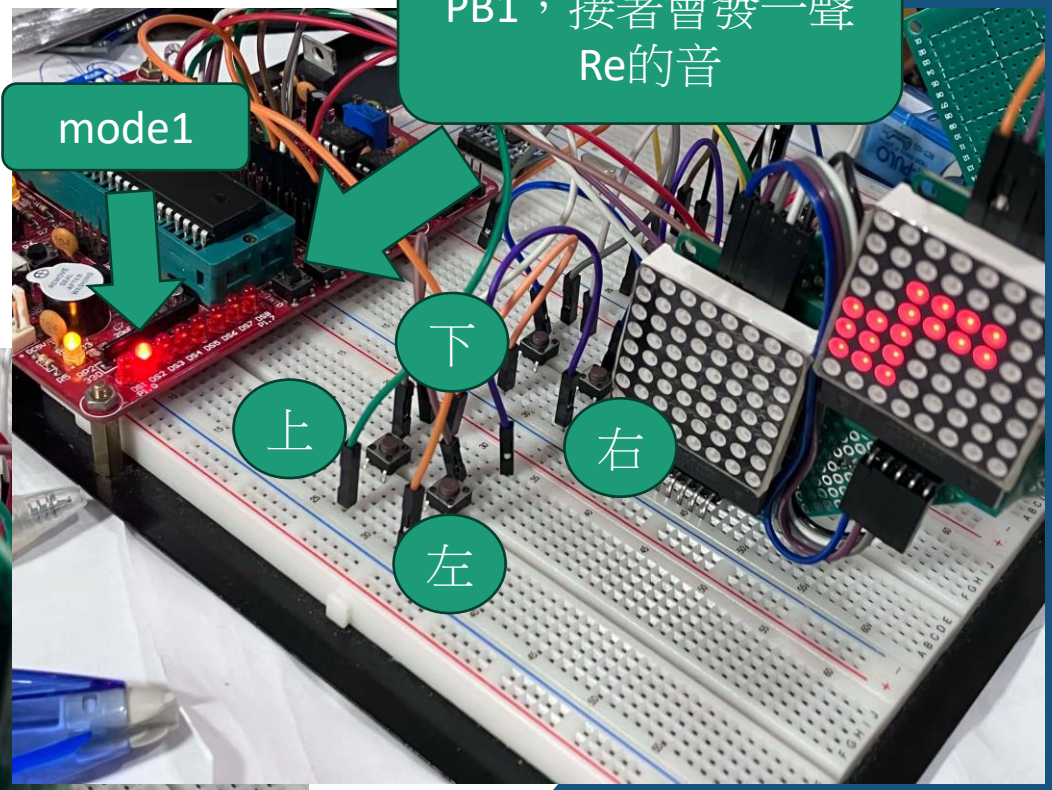
座標格式:
Column: 1-8
Row:0-7



EX: 2*4的軍艦
其左上角的座標:
col 3, row 2

設定好一個軍艦後按
PB1，接著會發一聲
Re的音

mode1



流程、設計: mode1

首先，程式會先幫玩家找到一個合法的位置。具體流程如下:

1. 枚舉每一個位置
2. 檢查是否碰到邊界
3. 檢查是否和前一個軍艦有所重疊

▼檢查重疊

```
450  char overlap(char x1, char y1, char h1, char w1,
451                    char x2, char y2, char h2, char w2){
452      for (i=x1;i<x1+h1;i++){
453          for (j=y1;j<y1+w1;j++){
454
455              if ((i >= x2) && (i < x2 + h2)){
456                  if ((j >= y2) && (j < y2+w2)){
457                      return 1;
458                  }
459              }
460          }
461      }return 0;
462  }
463
```

```
void mode1(void){
    OLED_Clear();
    OLED_SetCursor(2, 3);
    OLED_DisplayString("Player1 setting ");

    conti_flag = 0;
    nxt_row = 0; nxt_col = 0;
    check_ = 0;
    now_key;
    idx = 0; // 當前要處理的軍艦
    while (idx < 2){ // 有兩個軍艦要設置
        conti_flag = 0;
        for (tmp_l=0;tmp_l<idx;tmp_l++){ // 先找到一個合法位置，將它設為預設位置
            for (tmp_i=1;tmp_i<=8;tmp_i++){
                for (tmp_j=0;tmp_j<8;tmp_j++){
                    if (conti_flag != 0)break;;
                    if (tmp_j+boat_type[idx][1]-1 > 7 ){continue;} // 邊界檢查
                    if (tmp_i+boat_type[idx][0]-1 > 8){continue;}
                    if (overlap(tmp_i, tmp_j, boat_type[idx][0], boat_type[idx][1],
                                player1[tmp_l][0], player1[tmp_l][1], boat_type[tmp_l][0], boat_type[tmp_l][1]) != 1){
                                  // overlap 檢查和先前的軍艦是否有重疊
                        player1[idx][0] = tmp_i;
                        player1[idx][1] = tmp_j; // 將他們儲存
                        conti_flag = 1;
                        break;
                    }
                }
            }
```

流程、設計: mode1

接著才是由玩家控制，玩家控制上下左右按鍵(可以看PPT page 6)來移動軍艦。
每次按下按鍵時，程式先計算出下一個位置，同時做邊界修正。

接著在和當前點矩陣盤面的其他軍艦做對比，檢查是否有重疊。有的話(不合法)直接continue，重新偵測按鍵。

```
now_key = Get_Key();
if (now_key != 0){
    delay_ms(10);
    if (Get_Key() == now_key){ // 按鍵控制+邊界處理
        nxt_col = player1[idx][0];nxt_row = player1[idx][1];
        if (now_key == 1){// up r
            nxt_row = (player1[idx][1] <= 0)? player1[idx][1]:player1[idx][1]-1;
        }if (now_key == 2){ // down r
            nxt_row = (player1[idx][1]+boat_type[idx][1]-1 >= 7)? player1[idx][1]:player1[idx][1]+1;
        }
        if (now_key == 5){// left r
            nxt_col = (player1[idx][0] <= 1)? player1[idx][0]:player1[idx][0]-1;
        }
        if (now_key == 6){// right r
            nxt_col = (player1[idx][0]+boat_type[idx][0]-1 >= 8)? player1[idx][0]:player1[idx][0]+1;
        }// checking

        for (l=0;l<idx;l++){ // 檢查這個位置是否合法(有無重疊?)
            conti_flag = 0;
            if (overlap(nxt_col, nxt_row, boat_type[idx][0], boat_type[idx][1],
                        player1[l][0], player1[l][1], boat_type[l][0], boat_type[l][1]) == 1){
                conti_flag = 1;
                break;;
            }
        }
        if (conti_flag == 1){ // 不合法，繼續執行
            conti_flag = 0;
            continue;
        }
        /* for debug
```

流程、設計: mode1

如果合法，那刪除原本在點矩陣陣列裡儲存的點，接著畫上新區域的點。

處理完軍艦移動後，顯示陣列裡的資料在點矩陣上。這裡是顯示在第2個點矩陣上。

兩個軍艦都設定好後轉到mode2

```
            // 刪除原本軍艦在點矩陣上的位置
            del(player1[idx][0], player1[idx][1], player1[idx][0]+boat_type[idx][0]-1, player1[idx][1]+boat_type[idx][1]-1, 0);
            player1[idx][1] = nxt_row;
            player1[idx][0] = nxt_col;
            // 標記新的區域
            draw(player1[idx][0], player1[idx][1], player1[idx][0]+boat_type[idx][0]-1, player1[idx][1]+boat_type[idx][1]-1, 0);
            now_key = 0;
            delay_ms(100);
        }

    } // 顯示
    for (i=1;i<=8;i++){
        Writesingle7219(2, i, mat1_1[i]);
    } // 確定軍艦的位置
    if (but1 == 0){
        delay_ms(10);
        if (but1 == 0){
            idx+=1;
            check_ = 0;
            for(in_i = 0;in_i < 588;in_i++){
                P3_7 = 1;
                Delay_Re(); // 發出Re的音
                P3_7 = 0;
                Delay_Re();
            }
            while (but1 == 0);
        }
    }
}}
// 都設定完成-> mode2
mode = 2;
```

```
void del(char x1, char y1, char x2, char y2, char type){
    if (type == 0){// mat1_1
        for (i=x1;i<=x2;i++){
            for (j=y1;j<=y2;j++){
                mask = 1<<j;
                mask = ~mask;
                mat1_1[i] &= mask;
            }
        }
    }
    if (type == 1){// mat2_2
        for (i=x1;i<=x2;i++){
            for (j=y1;j<=y2;j++){
                mask = 1<<j;
                mask = ~mask;
                mat2_2[i] &= mask;
            }
        }
    }
}
```

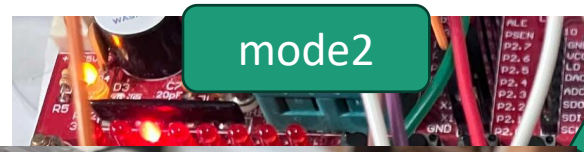
```
void draw(char x1, char y1, char x2, char y2, char type
    if (type == 0){// mat1_1
        for (i=x1;i<=x2;i++){
            for (j=y1;j<=y2;j++){
                mask = 1<<j;
                mat1_1[i] |= mask;
            }
        }
    }
    if (type == 1){// mat1_1
        for (i=x1;i<=x2;i++){
            for (j=y1;j<=y2;j++){
                mask = 1<<j;
                mat2_2[i] |= mask;
            }
        }
    }
    if (type == 2){// mat1_2
        for (i=x1;i<=x2;i++){
            for (j=y1;j<=y2;j++){
                mask = 1<<j;
                mat1_2[i] |= mask;
```

流程、設計: mode2

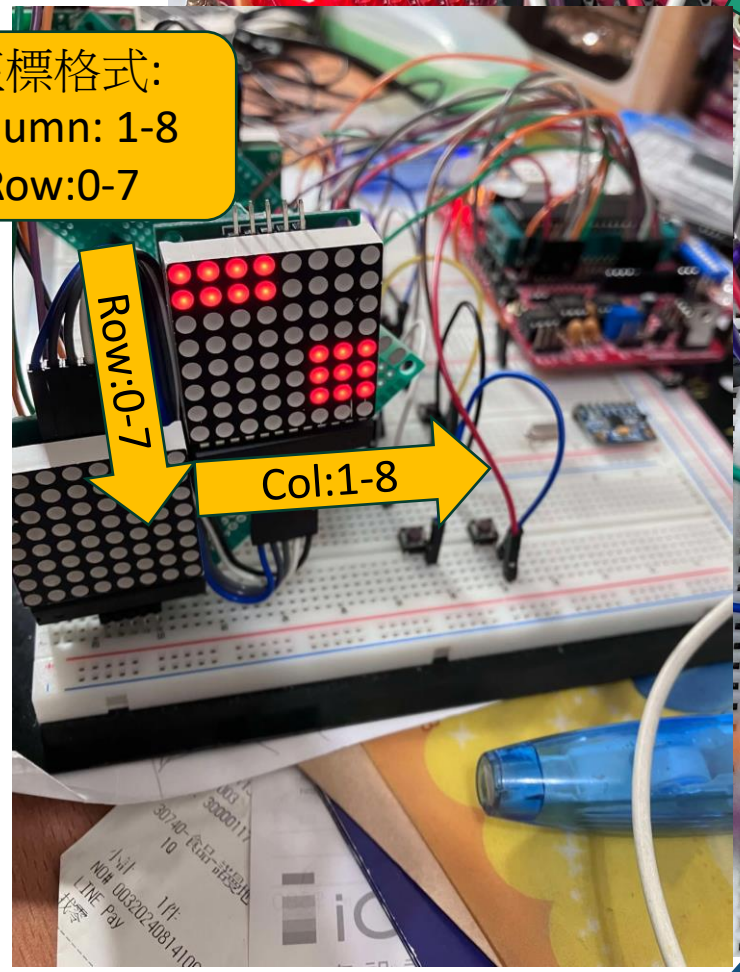
顯示在設定哪一個軍艦:
Setting 0: 設定2*4的軍艦
Setting 1: 設定3*3的軍艦

下面兩行顯示現在控制的軍艦其左上角的座標

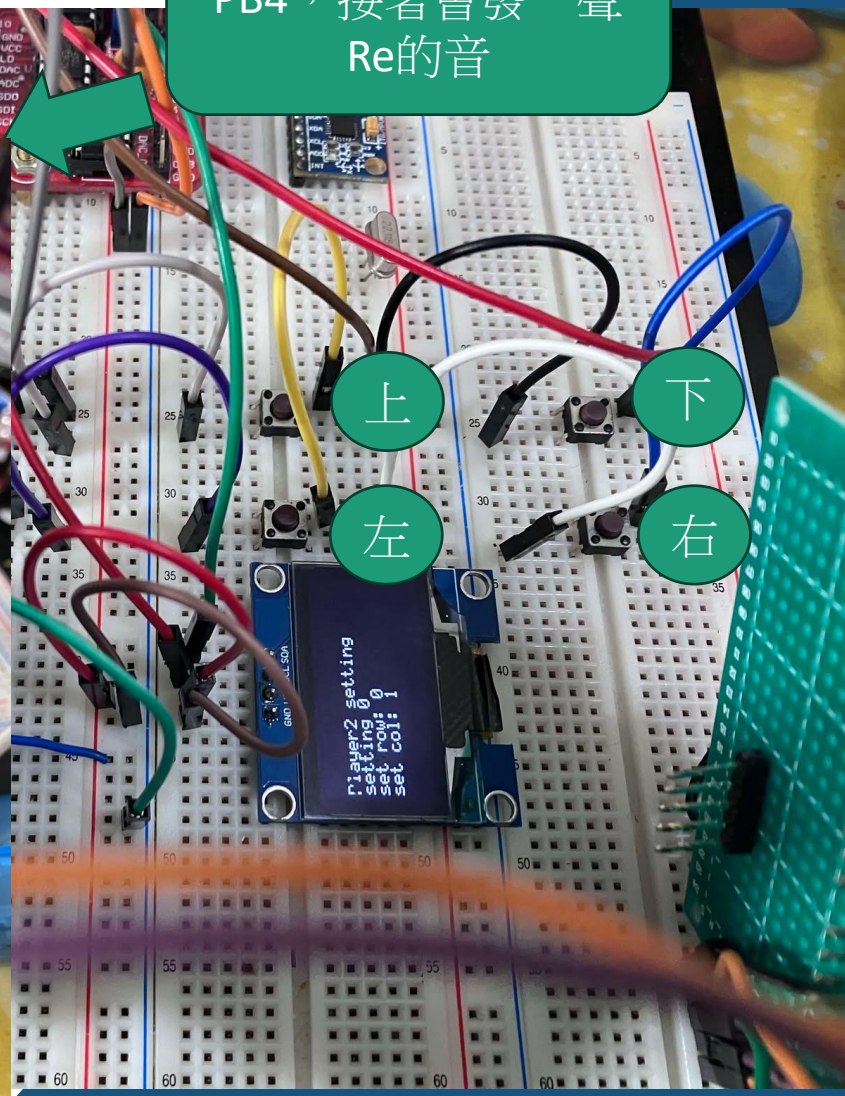
玩家2 的回合
放置軍艦



座標格式:
Column: 1-8
Row:0-7



設定好一個軍艦後按PB4，接著會發一聲Re的音



流程、設計: mode2

這裡程式的部分跟mode1一樣，只是把player1的資料改成player2的

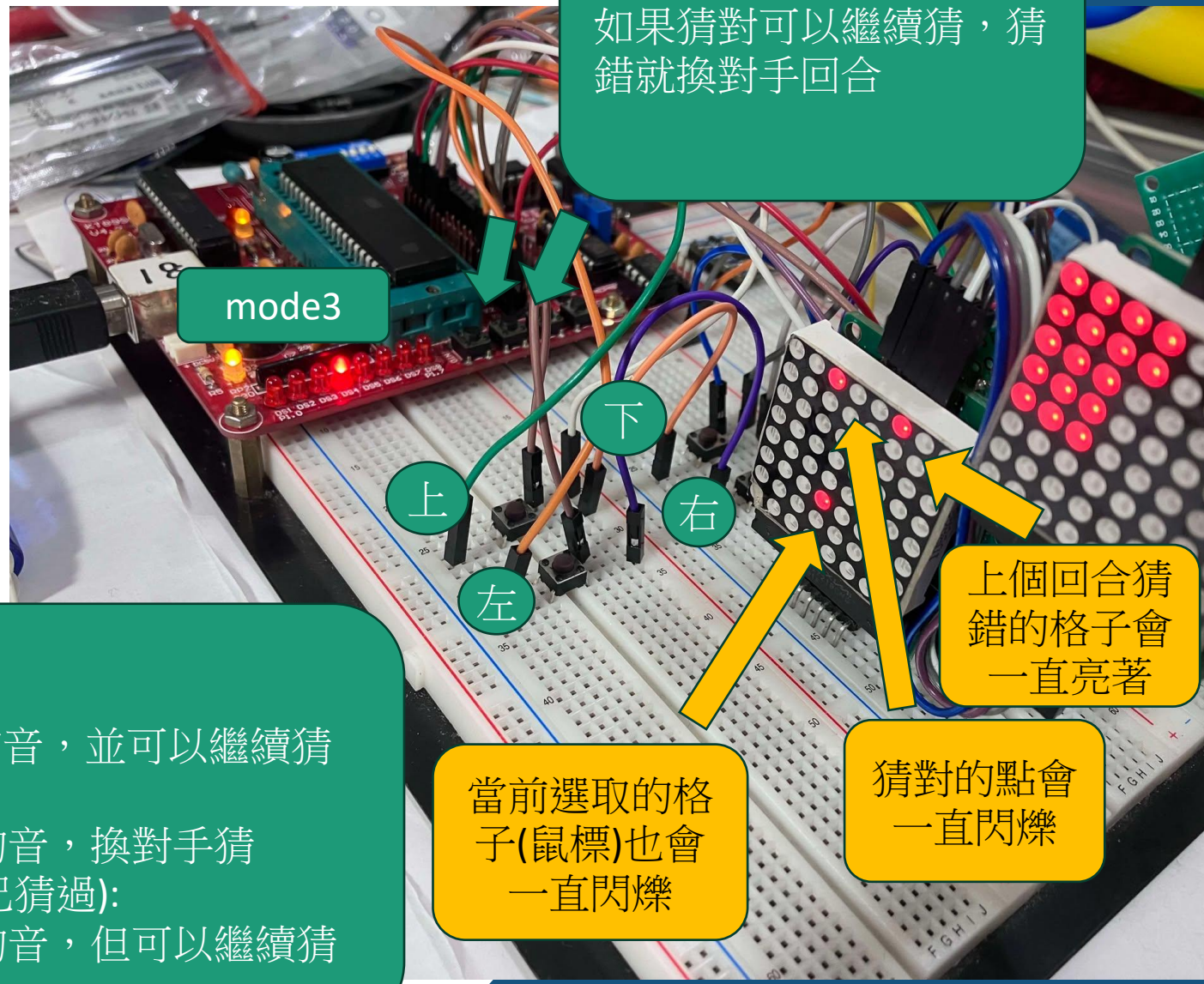
```
void mode2(void){
    OLED_Clear();
    OLED_SetCursor(2, 3);
    OLED_DisplayString("Player2 setting ");

    conti_flag = 0;
    // col (1 ~ 8 (left to right)), row(0 ~ 7(up to down))
    nxt_row = 0; nxt_col = 0;
    check_ = 0;
    idx = 0;
    while (idx < 2){
        conti_flag = 0;
        for (tmp_l=0;tmp_l<idx;tmp_l++){ // default idx
            for (tmp_i=1;tmp_i<=8;tmp_i++){
                for (tmp_j=0;tmp_j<8;tmp_j++){
                    if (conti_flag != 0)break;;
                    if (tmp_j+boat_type[idx][1]-1 > 7 ){continue;}
                    if (tmp_i+boat_type[idx][0]-1 > 8){continue;}
                    // 這裡改成player2的陣列
                    if (overlap(tmp_i, tmp_j, boat_type[idx][0], boat_type[idx][1],
                                player2[tmp_l][0], player2[tmp_l][1], boat_type[tmp_l][0], boat_type[tmp_l][1]) != 1){
                        player2[idx][0] = tmp_i;
                        player2[idx][1] = tmp_j;
                        conti_flag = 1;
                        break;
                    }
                }
            }
        }
```

```
        while (check_){
            OLED_SetCursor(4, 3);
            OLED_DisplayString("set row: ");
            OLED_DisplayChar('0' + player2[idx][1]);
            OLED_SetCursor(5, 3);
            OLED_DisplayString("set col: ");
            OLED_DisplayChar('0' + player2[idx][0]);


            now_key = Get_Key();
            if (now_key != 0){
                delay_ms(10);
                if (Get_Key() == now_key){
                    nxt_col = player2[idx][0];nxt_row = player2[idx][1];
                    if (now_key == 3){// up r
                        nxt_row = (player2[idx][1] <= 0)? player2[idx][1]:player2[idx][1]-1;
                    }if (now_key == 4){ // down r
                        nxt_row = (player2[idx][1]+boat_type[idx][1]-1 >= 7)? player2[idx][1]:player2[idx][1]+1;
                    }
                    if (now_key == 7){// left r
                        nxt_col = (player2[idx][0] <= 1)? player2[idx][0]:player2[idx][0]-1;
                    }
                    if (now_key == 8){// right r
                        nxt_col = (player2[idx][0]+boat_type[idx][0]-1 >= 8)? player2[idx][0]:player2[idx][0]+1;
                    }// checking

                    for (l=0;l<idx;l++){
                        conti_flag = 0;
                        if (overlap(nxt_col, nxt_row, boat_type[idx][0], boat_type[idx][1],
                                player2[l][0], player2[l][1], boat_type[l][0], boat_type[l][1]) == 1){
                            conti_flag = 1;
                            break;;
                        }
                    }
                    if (conti_flag == 1){
                        conti_flag = 0;
                        continue;
                    }
```

```
301                     del(player2[idx][0], player2[idx][1], player2[idx][0]+boat_type[idx][0]-1, player2[idx][1]+boat_type[idx][1]-1, 1);
302                     player2[idx][1] = nxt_row;
303                     player2[idx][0] = nxt_col;
304                     draw(player2[idx][0], player2[idx][1], player2[idx][0]+boat_type[idx][0]-1, player2[idx][1]+boat_type[idx][1]-1, 1
305                     now_key = 0;
306                     delay_ms(100);
307                 }
308
309             }
310             for (i=1;i<=8;i++){
311                 Writesingle7219(4, i, mat2_2[i]);
312             }
313             if (but4 == 0){
314                 delay_ms(10);
315                 if (but4 == 0){
316                     idx+=1;
317                     check_ = 0;
318                     for(in_i = 0;in_i < 588;in_i++){
319                         P3_7 = 1;
320                         Delay_Re();
321                         P3_7 = 0;
322                         Delay_Re();
323                     }
324                     while (but4 == 0);
325                 }
326             }
327
328         }
329         }
330         mode = 3;
331     }
332
```

流程、設計: mode3

Player1 猜的回合
當前座標
當前得分(滿分共17分)



按PB1鍵是跳過自己回合
按PB2鍵是猜格子
如果猜對可以繼續猜，猜錯就換對手回合

上個回合猜錯的格子會一直亮著

猜對的點會一直閃爍

當前選取的格子(鼠標)也會一直閃爍

猜對:
蜂鳴器發Re的音，並可以繼續猜
猜錯:
蜂鳴器發Do的音，換對手猜
猜重複格子(已猜過):
蜂鳴器發Do的音，但可以繼續猜

流程、設計: mode3

要顯示在OLED上面的文字。

這裡先把要打的文字在迴圈外就打上，實際的數字在迴圈裡顯示。減少8051要處理的指令

```
333  void mode3(void){
334      turn = 1;
335      OLED_Clear();
336      OLED_SetCursor(2, 3);
337      OLED_DisplayString("Player ");
338      OLED_DisplayChar('0'+turn);
339      OLED_DisplayString(" turn ");
340      OLED_SetCursor(3, 4);
341      OLED_DisplayString("now row: ");
342      OLED_SetCursor(4, 4);
343      OLED_DisplayString("now col: ");
344      OLED_SetCursor(5, 4);
345      OLED_DisplayString("now score: "); // OLED要顯示的資訊
346
347
348      while (winner == 0){ // 若還沒產生贏家
349
350
351          now_col = player[turn][0];
352          now_row = player[turn][1];
353          OLED_SetCursor(3, 55);
354          OLED_DisplayChar('0'+now_row);
355          OLED_SetCursor(4, 55);
356          OLED_DisplayChar('0'+now_col);
357          OLED_SetCursor(5, 70);
358          OLED_DisplayChar(player_score[turn]/10+'0');
359          OLED_DisplayChar(player_score[turn]%10 + '0'); // 顯示當前資訊
```

流程、設計: mode3

鼠標的移動
每次移動完會做一個紀錄。下次回到自己的回合時會停留在上次鼠標的位置。

按下PB1的程式

```
now_key = Get_Key();
if (now_key != 0){
    delay_ms(10);
    if (now_key == Get_Key()){ //  移動鼠標
        nxt_col = now_col;nxt_row = now_row;
        if (now_key == 1){
            nxt_row = (now_row <= 0)? 0:now_row-1;
        }else if (now_key == 2){
            nxt_row = (now_row >= 7)? now_row:now_row+1;
        }else if (now_key == 5){
            nxt_col = (now_col <= 1)? 1:now_col-1;
        }else if (now_key == 6){
            nxt_col = (now_col >= 8)? now_col:now_col+1;
        }
        now_col = nxt_col;
        now_row = nxt_row;
        player[turn][0] = now_col;
        player[turn][1] = now_row;
    }
}

if (but1 == 0){ // 按下but1，跳過自己回合
    delay_ms(10);
    if (but1 == 0){
        player[turn][0] = now_col;
        player[turn][1] = now_row;
        mode = 4;
        break;
    }
}
```

流程、設計: mode3

按下PB2，猜格子的程式碼

猜到重複的或猜錯會響Do的音
猜對響Re的音

猜對後會判斷是否達勝利條件

若達到勝利條件會設定winner的值。若winner是0就是指還沒出現贏家。否則winner值應為1或2

```
390         if (but2 == 0){ // 按but2，猜格子
391             delay_ms(10);
392             if (but2 == 0){
393                 if (Get_Dot(now_col, now_row, 2) != 0){ // 猜重複格子
394                     for(in_i = 0;in_i < 523;in_i++){
395                         P3_7 = 1;
396                         Delay_Do();
397                         P3_7 = 0;
398                         Delay_Do();
399                     }
400                     continue;
401                 }
402                 draw(now_col, now_row, now_col, now_row, 2); // 記錄這一點有猜過
403                 if ((Get_Dot(now_col, now_row, 1) == 0)){    // 猜錯
404                     for(in_i = 0;in_i < 523;in_i++){
405                         P3_7 = 1;
406                         Delay_Do();
407                         P3_7 = 0;
408                         Delay_Do();
409                     }
410                     player[turn][0] = now_col;
411                     player[turn][1] = now_row;
412                     mode = 4; // 跳出mode3 迴圈，換mode4 player2 猜
413                     break;
414                 }else{
415                     for(in_i = 0;in_i < 588;in_i++){// 猜對
416                         P3_7 = 1;
417                         Delay_Re();
418                         P3_7 = 0;
419                         Delay_Re();
420                     }
421                     player_score[turn]+=1;
422                     if (player_score[turn] >= 17){ // 判斷自己有沒有贏
423                         winner = turn;
424                         break;
```

流程、設計: mode3

顯示資訊在點矩陣顯示器上。
透過延遲0.1秒來達到閃爍的效果

用mask和位元控制來把當前鼠標位置和猜對位置的led燈熄熄滅

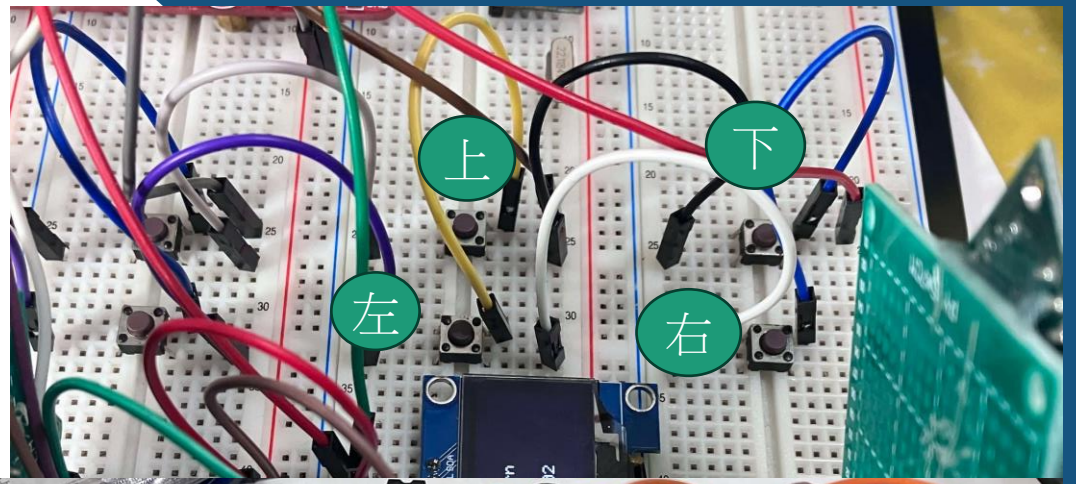
```
430         for (i=1;i<=8;i++){
431             if (i == now_col)mask = 1 << now_row; // 記得顯示當前鼠標的位置
432             else mask = 0;
433             Writesingle7219(1, i, (mat1_2[i] | mask));
434         }delay_ms(100);
435         for (i=1;i<=8;i++){
436             mask = 0;
437             if (now_col == i)mask |= 1<<now_row; // 鼠標的位置
438             else mask = 0;
439             mask |= (mat2_2[i] & mat1_2[i]);     // 實際猜對的位置
440             mask = ~mask;
441             Writesingle7219(1, i, mat1_2[i]&mask); // 將上述的位置的燈熄滅，以達到閃爍的效果
442         }
443     }
444     if (winner != 0){
445         mode = 5;
446     }
447
```

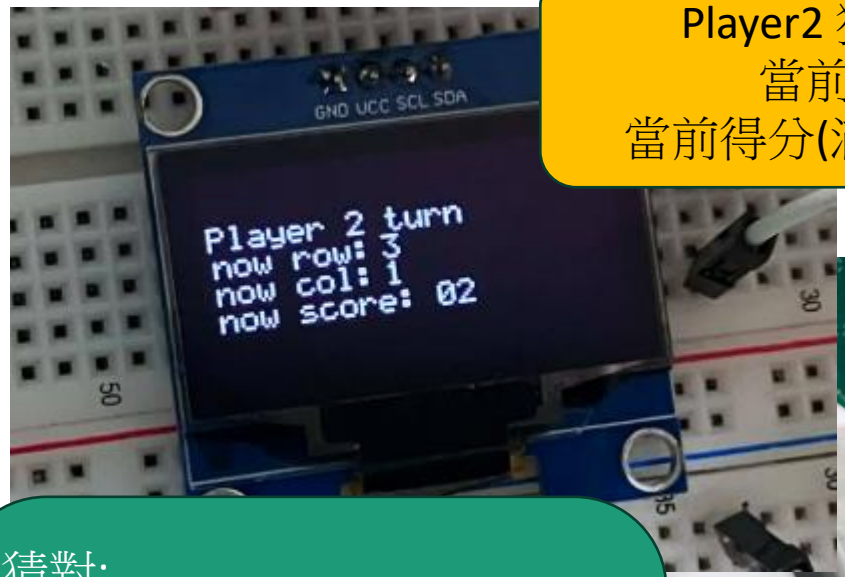
流程、設計: mode3

Get_Dot 來獲取特定座標格子的資料
(檢查該點是1還是0)

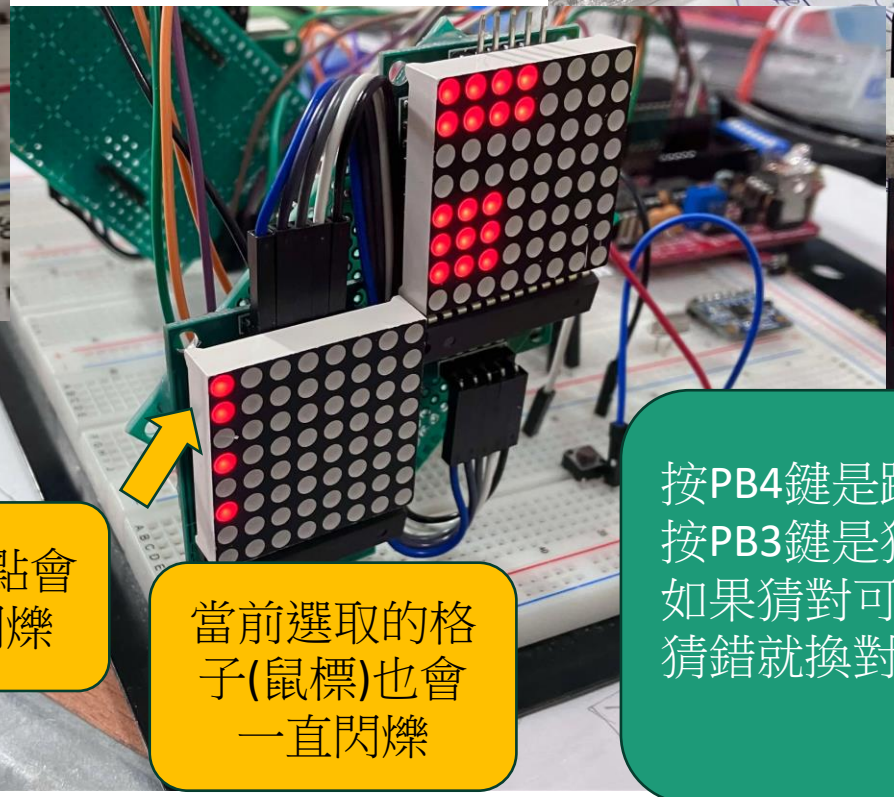
```
519 char Get_Dot(char col, char row,char type){
520
521     if (type == 0){// mat1_1
522         mask = 1<<row;
523         get_bit = mask & mat1_1[col];
524         return get_bit;
525     }
526     else if (type == 1){// mat2_2
527         mask = 1<<row;
528         get_bit = mask & mat2_2[col];
529         return get_bit;
530     }
531     else if (type == 2){// mat1_2
532         mask = 1<<row;
533         get_bit = mask & mat1_2[col];
534         return get_bit;
535     }
536     else if (type == 3){// mat2_1
537         mask = 1<<row;
538         get_bit = mask & mat2_1[col];
539         return get_bit;
540     }
541     return 0;
542 }
543
```

流程、設計: mode4



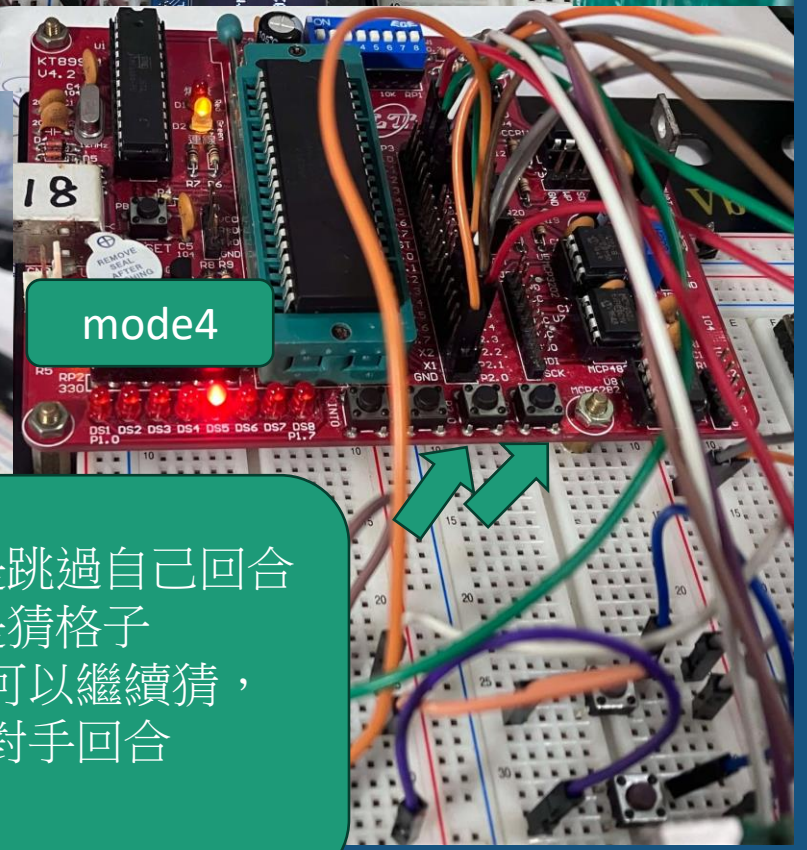


Player2 猜的回合
當前座標
當前得分(滿分共17分)



猜對的點會一直閃爍

當前選取的格子(鼠標)也會一直閃爍



猜對:
蜂鳴器發Re的音，並可以繼續猜
猜錯:
蜂鳴器發Do的音，換對手猜
猜重複格子(已猜過):
蜂鳴器發Do的音，但可以繼續猜

按PB4鍵是跳過自己回合
按PB3鍵是猜格子
如果猜對可以繼續猜，
猜錯就換對手回合

流程、設計: mode4

這裡程式的部分跟mode3一樣，只是把player1的資料改成player2的

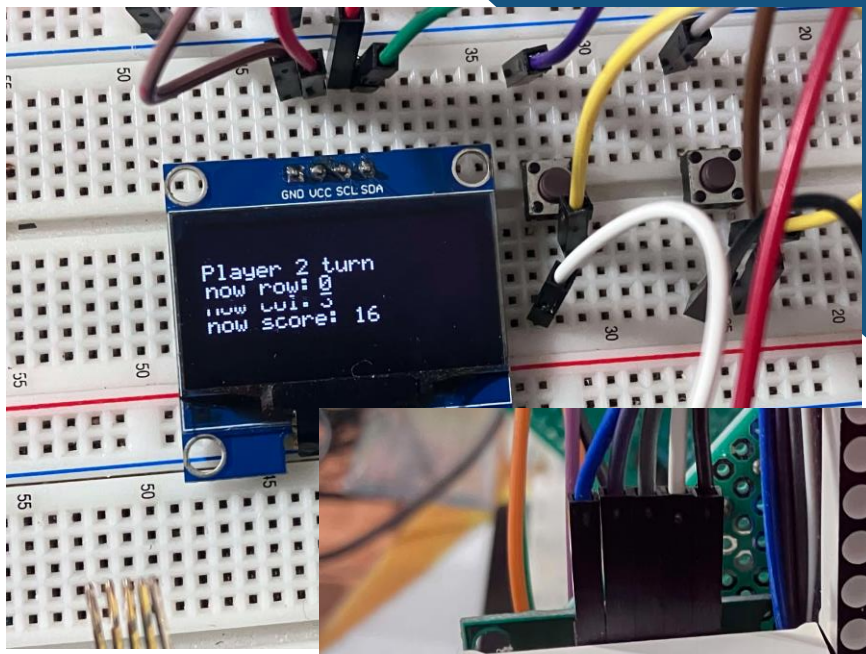
```
589         if (but4 == 0){
590             delay_ms(10);
591             if (but4 == 0){
592                 player[turn][0] = now_col;
593                 player[turn][1] = now_row;
594                 mode = 3;
595                 break;
596             }
597         }
598         if (but3 == 0){
599             delay_ms(10);
600             if (but3 == 0){
601                 if (Get_Dot(now_col, now_row, 3) != 0){
602                     for(in_i = 0;in_i < 523;in_i++){
603                         P3_7 = 1;
604                         Delay_Do();
605                         P3_7 = 0;
606                         Delay_Do();
607                     }
608                     continue;
609                 }
610                 draw(now_col, now_row, now_col, now_row, 3);
611                 if ((Get_Dot(now_col, now_row, 0) == 0)){
612                     for(in_i = 0;in_i < 523;in_i++){
613                         P3_7 = 1;
614                         Delay_Do();
615                         P3_7 = 0;
616                         Delay_Do();
617                     }
618                     player[turn][0] = now_col;
619                     player[turn][1] = now_row;
620                     mode = 3;
621                     break;
622                 }else{
623                     for(in_i = 0;in_i < 588;in_i++){
```

```
622                 }else{
623                     for(in_i = 0;in_i < 588;in_i++){// 猜對
624                         P3_7 = 1;
625                         Delay_Re();
626                         P3_7 = 0;
627                         Delay_Re();
628                     }
629                     player_score[turn]+=1;
630                     if (player_score[turn] >= 17){
631                         winner = turn;
632                         break;
633                     }
634                 }
635             }
636         }
637         for (i=1;i<=8;i++){
638             mask = 0;
639             if (i == now_col)mask = 1 << now_row;
640             else mask = 0;
641             Writesingle7219(3, i, (mat2_1[i] | mask));
642         }delay_ms(100);
643         for (i=1;i<=8;i++){
644             mask = 0;
645             if (now_col == i)mask |= 1<<now_row;
646             else mask = 0;
647             mask |= (mat1_1[i] & mat2_1[i]);
648             mask = ~mask;
649             Writesingle7219(3, i, mat2_1[i]&mask);
650         }
651     }
652     if (winner != 0){
653         mode = 5;
654     }
655
```

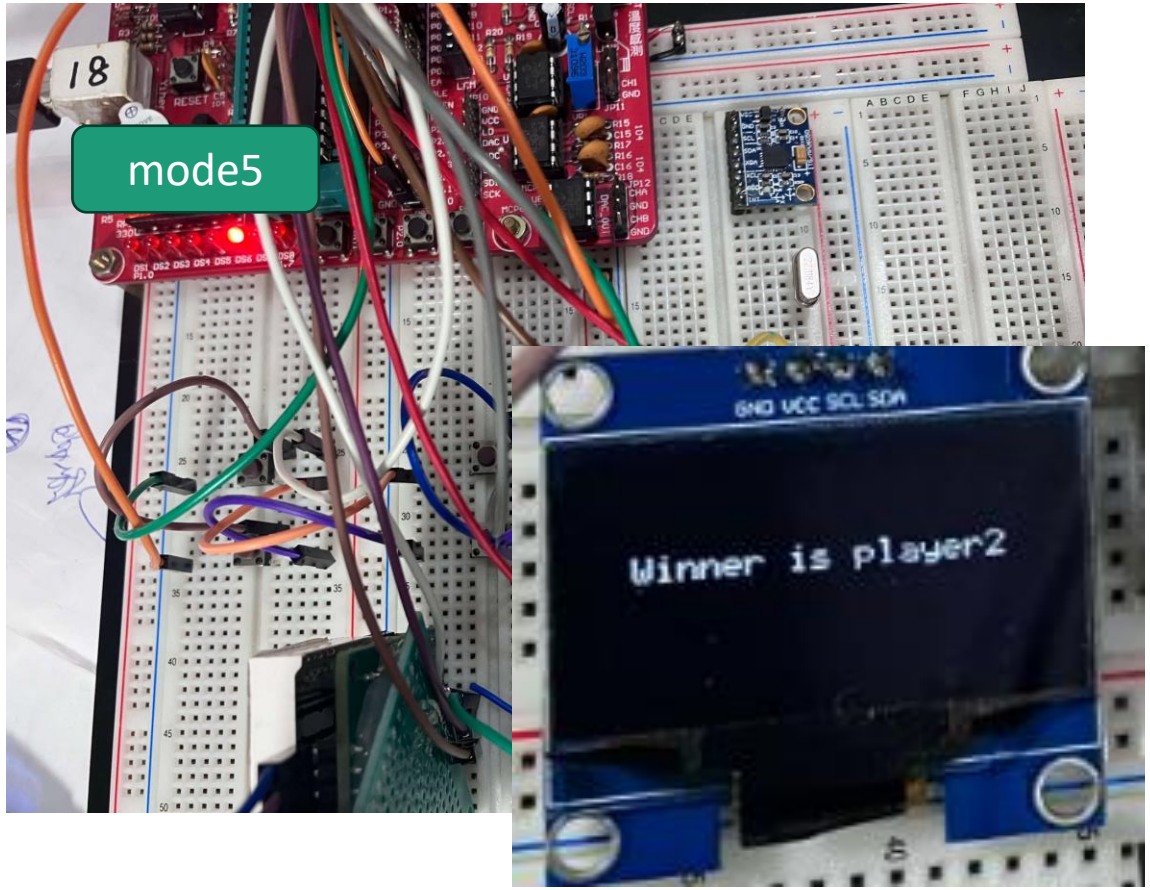
```
544 void mode4(void){
545     turn = 2;
546     OLED_Clear();
547     OLED_SetCursor(2, 3);
548     OLED_DisplayString("Player ");
549     OLED_DisplayChar('0'+turn);
550     OLED_DisplayString(" turn ");
551     OLED_SetCursor(3, 4);
552     OLED_DisplayString("now row: ");
553     OLED_SetCursor(4, 4);
554     OLED_DisplayString("now col: ");
555
556
557     while (winner == 0){
558
559         now_col = player[turn][0];
560         now_row = player[turn][1];
561         OLED_SetCursor(3, 55);
562         OLED_DisplayChar('0'+now_row);
563         OLED_SetCursor(4, 55);
564         OLED_DisplayChar('0'+now_col);
565         OLED_SetCursor(5, 70);
566         OLED_DisplayChar(player_score[turn]/10+'0');
567         OLED_DisplayChar(player_score[turn]%10 + '0');
568         now_key = Get_Key();
569         if (now_key != 0){
570             delay_ms(10);
571             if (now_key == Get_Key()){
572                 nxt_col = now_col;nxt_row = now_row;
573                 if (now_key == 3){
574                     nxt_row = (now_row <= 0)? 0:now_row-1;
575                 }else if (now_key == 4){
576                     nxt_row = (now_row >= 7)? now_row:now_row+1;
577                 }else if (now_key == 7){
578                     nxt_col = (now_col <= 1)? 1:now_col-1;
579                 }else if (now_key == 8){
580                     nxt_col = (now_col >= 8)? now_col:now_col+1;
581                 }
582                 now_col = nxt_col;
583                 now_row = nxt_row;
584                 player[turn][0] = now_col;
585                 player[turn][1] = now_row;
586             }
587         }
```

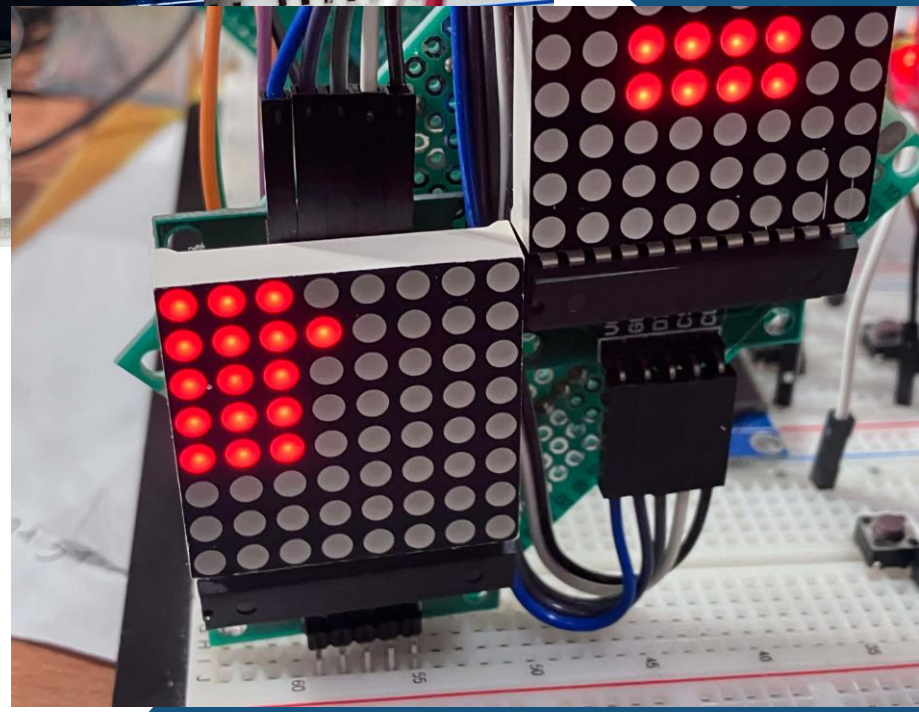
流程、設計: mode5

勝利前的畫面



勝利畫面





流程、設計: mode5

勝利之後就留在mode5的狀態
若要重新一把就按下reset 鍵

```
659  void mode5(void){
660      OLED_Clear();
661      OLED_SetCursor(3, 13);
662      OLED_DisplayString("Winner is player");
663      OLED_DisplayChar('0'+winner);
664      while (1);
665  }
666
667
```

Demo Video

https://drive.google.com/file/d/1zmNALnUtqhiJQWjD7kk94YrMHacoNfCG/view?usp=sharing

Github link:
https://github.com/rickyC3/MicroProcessor-Final-Project

報告到這裡結束，謝謝教授和助教

PS: 有一塊點矩陣顯示器和OLED有壞，還請助教操作時記得更換