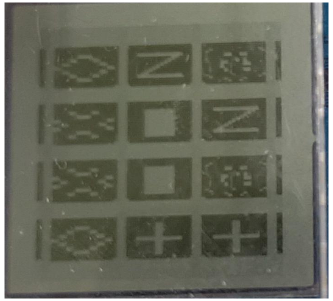
記憶大考驗-對對碰

309512074黃柏叡
310512009　陳懿
310512025賴知瑜

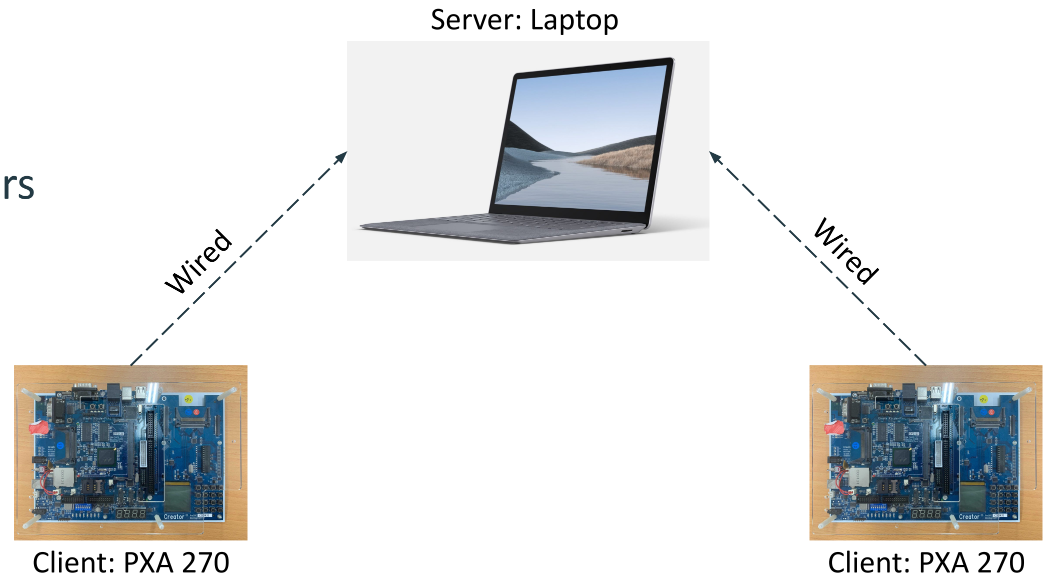
遊戲簡介



- 製作一個多人連線遊戲'記憶大考驗-對對碰'', 回合制
- PXA270的LCD顯示板上面會顯示4x3的圖案, 共6種圖案(兩兩一組), 接著消失, 參賽者有10秒的時間將圖案配對成功, 每成功配對一組圖形將得到1分(圖案與PXA270按鍵一對一對應), 並將配對成功的組合印出, 若按錯則圖形不會顯示
- 參賽者之間的LCD顯示的圖形是連動的, 假如A參賽者將第一組圖案配對成功, 則B參賽者就無法配對第一組圖案(同時第一組圖案也會在B的LCD面板上顯示), 因此這個遊戲不僅要考驗記憶力, 也考驗參賽者的速度
- 參賽者的分數會分別顯示在7段顯示器上以及記錄在Server上, 同時會用LED燈來提醒參賽者剩餘的回合數, 所有回合結束分數高者即獲勝

系統架構 - 硬體

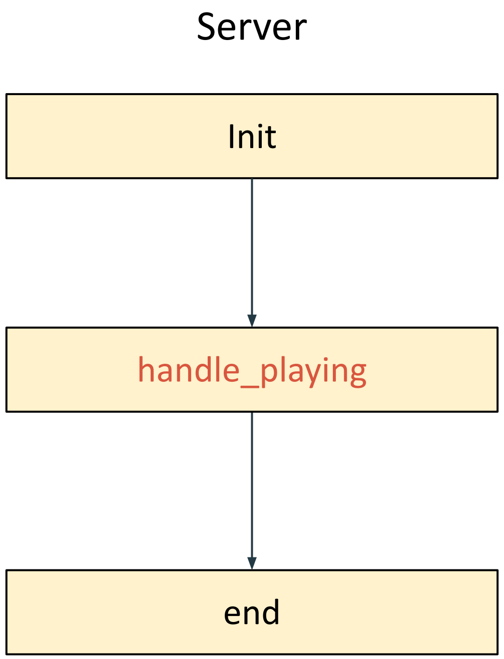
- Game Server
- PXA Client(1~N)
  - default: 2 players
  - support multiple players

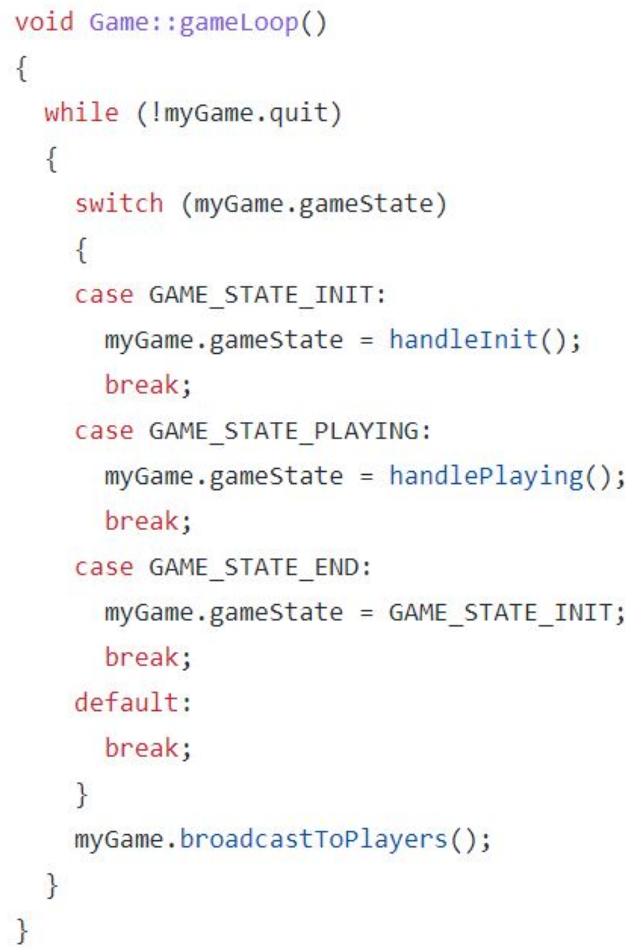


系統架構 - 軟體

- `client.cpp`: Main function to run the game on PXA270.
- `data_utils.h`: Game variables for both client and server, structure for messages send from server.
- `game_client.cpp`: Functions for running client on PXA270 include: `readServer()`, `sendServer()`, `run()`, `read_pad()`, `show_LCD_pic()` and so on.
- `game.cpp`: Server, to control how this game work.
- `LCD.c`: Functions to visualize patterns or words on PXA270
- `LCD.h`: Saved patterns for `Memory Game`, include 6 patterns and one back ground (cards all hidden).
- `random_map.c`: Functions for generating the random patterns for the game.
- `main.cpp`: Main function to run the server.

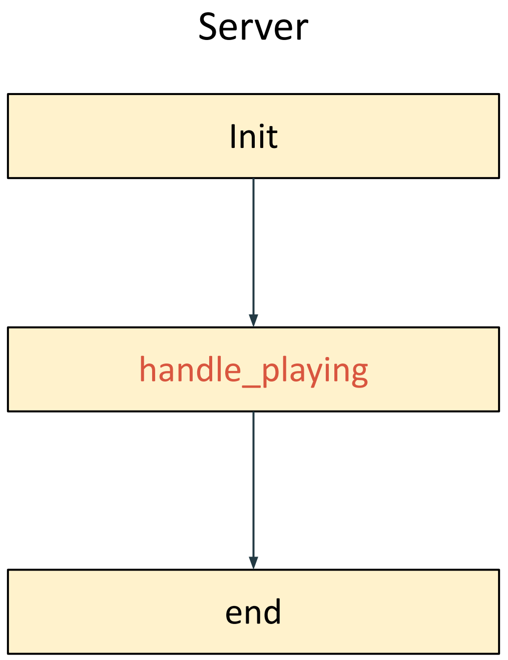
系統架構 - 流程圖

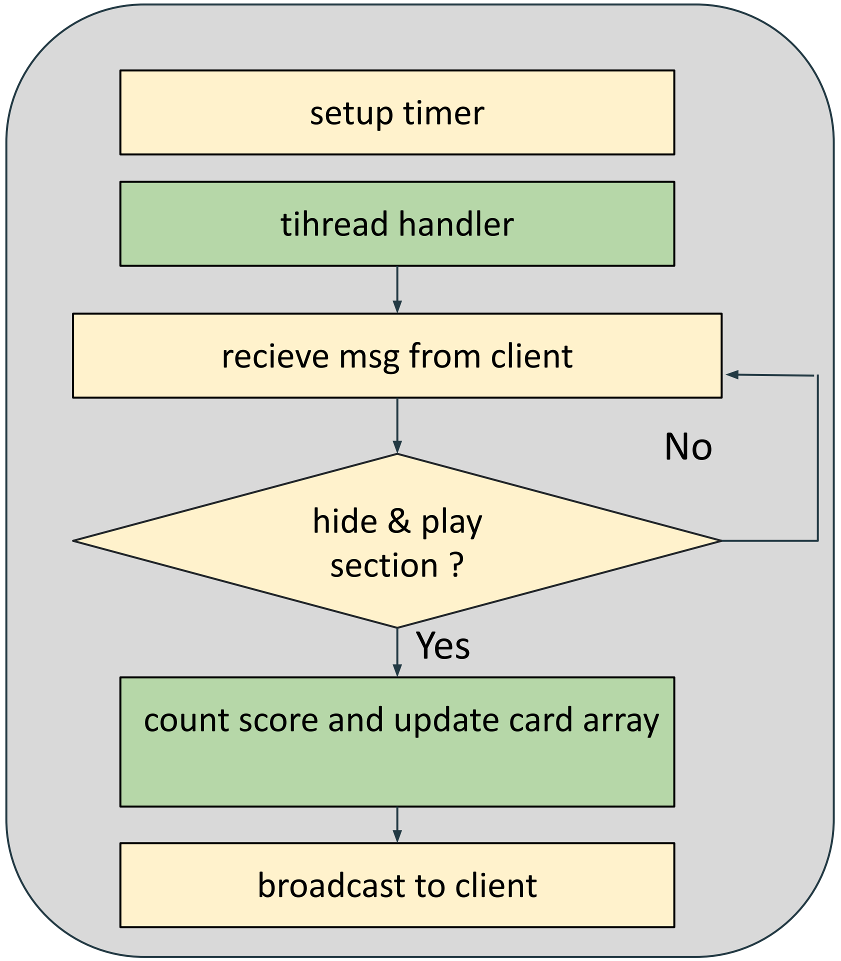




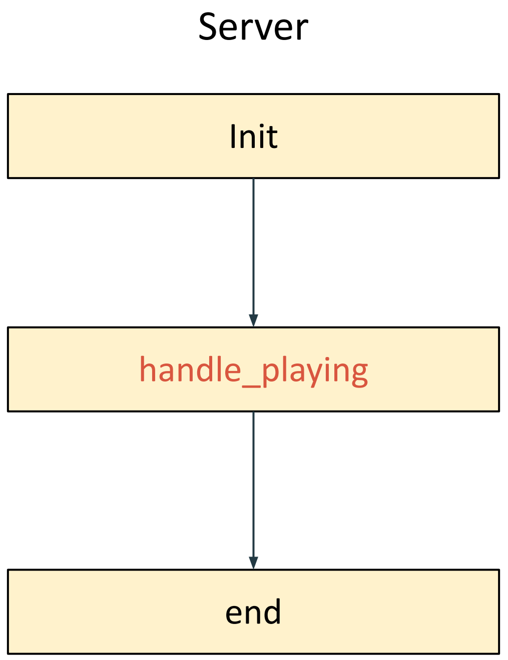
```
void Game::gameLoop()
{
  while (!myGame.quit)
  {
    switch (myGame.gameState)
    {
    case GAME_STATE_INIT:
      myGame.gameState = handleInit();
      break;
    case GAME_STATE_PLAYING:
      myGame.gameState = handlePlaying();
      break;
    case GAME_STATE_END:
      myGame.gameState = GAME_STATE_INIT;
      break;
    default:
      break;
    }
    myGame.broadcastToPlayers();
  }
}
```

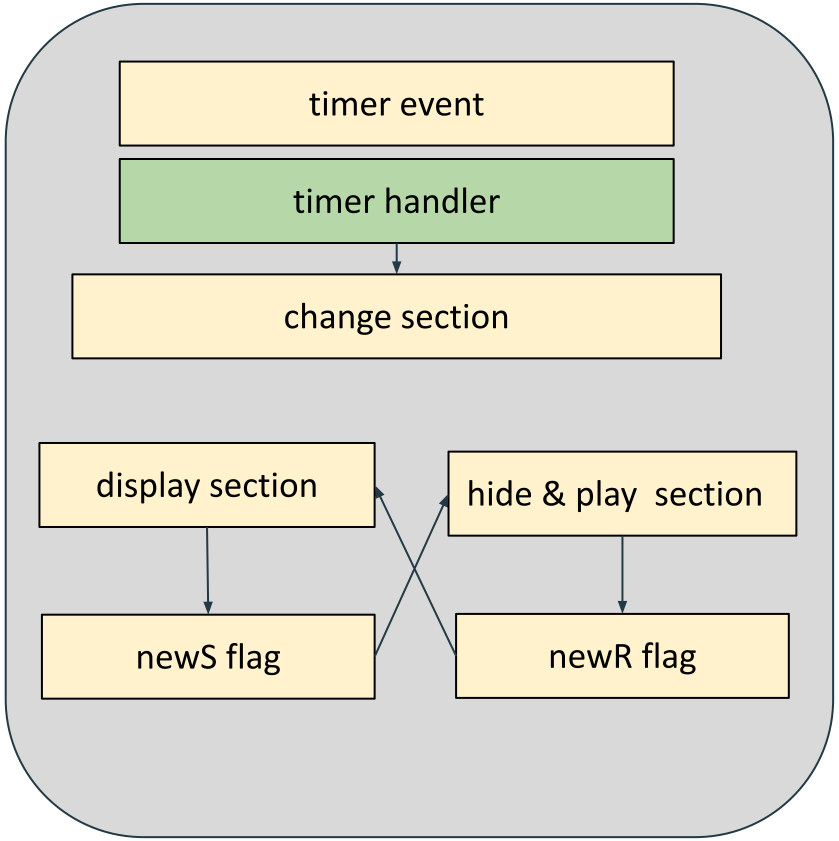
系統架構 - 流程圖


Server
Init
handle_playing
end

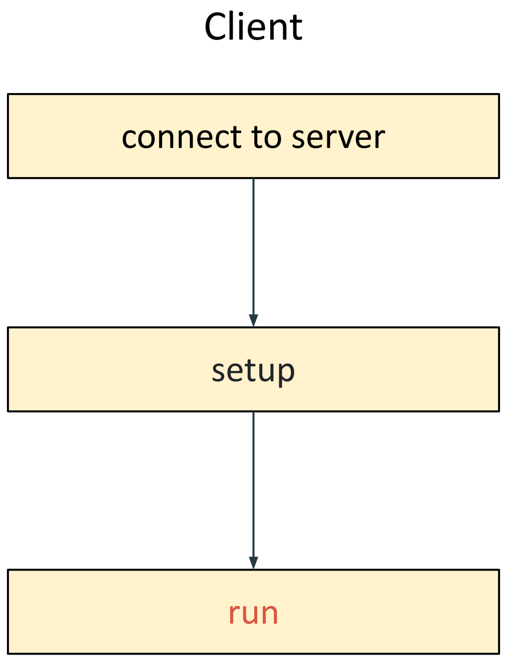

setup timer
tihread handler
recieve msg from client
hide & play section ?
No
Yes
count score and update card array
broadcast to client

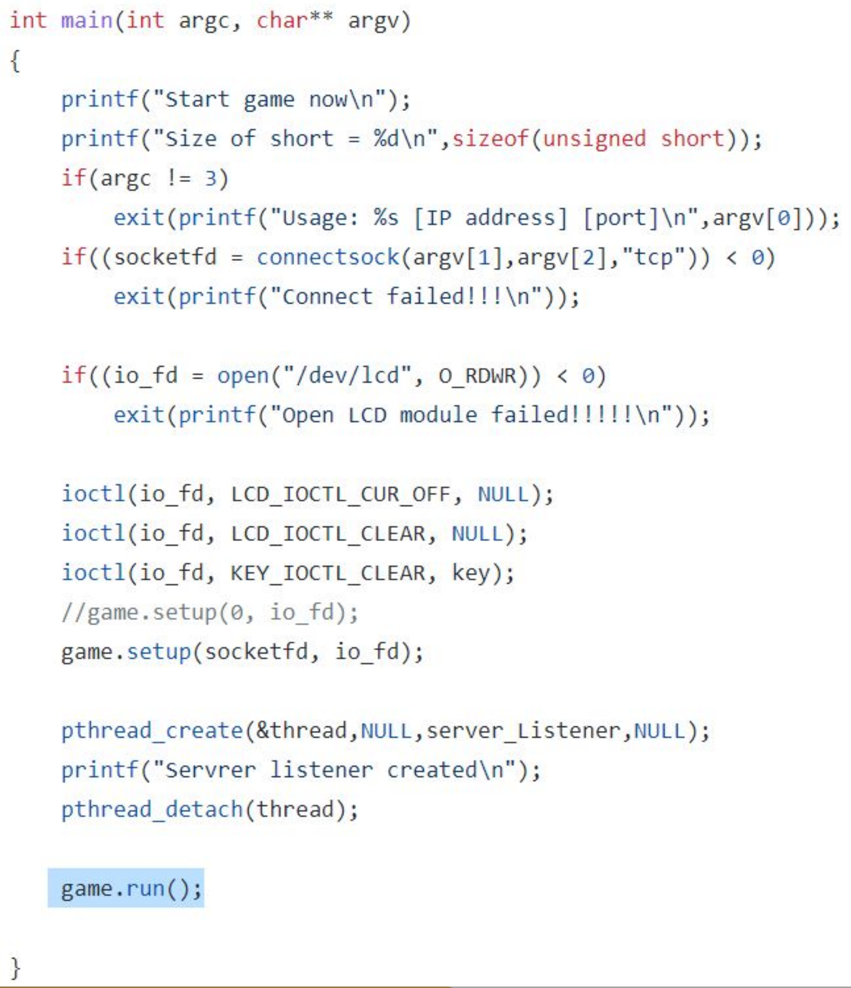
系統架構 - 流程圖


Server
Init
handle_playing
end


timer event
timer handler
change section
display section
hide & play section
newS flag
newR flag

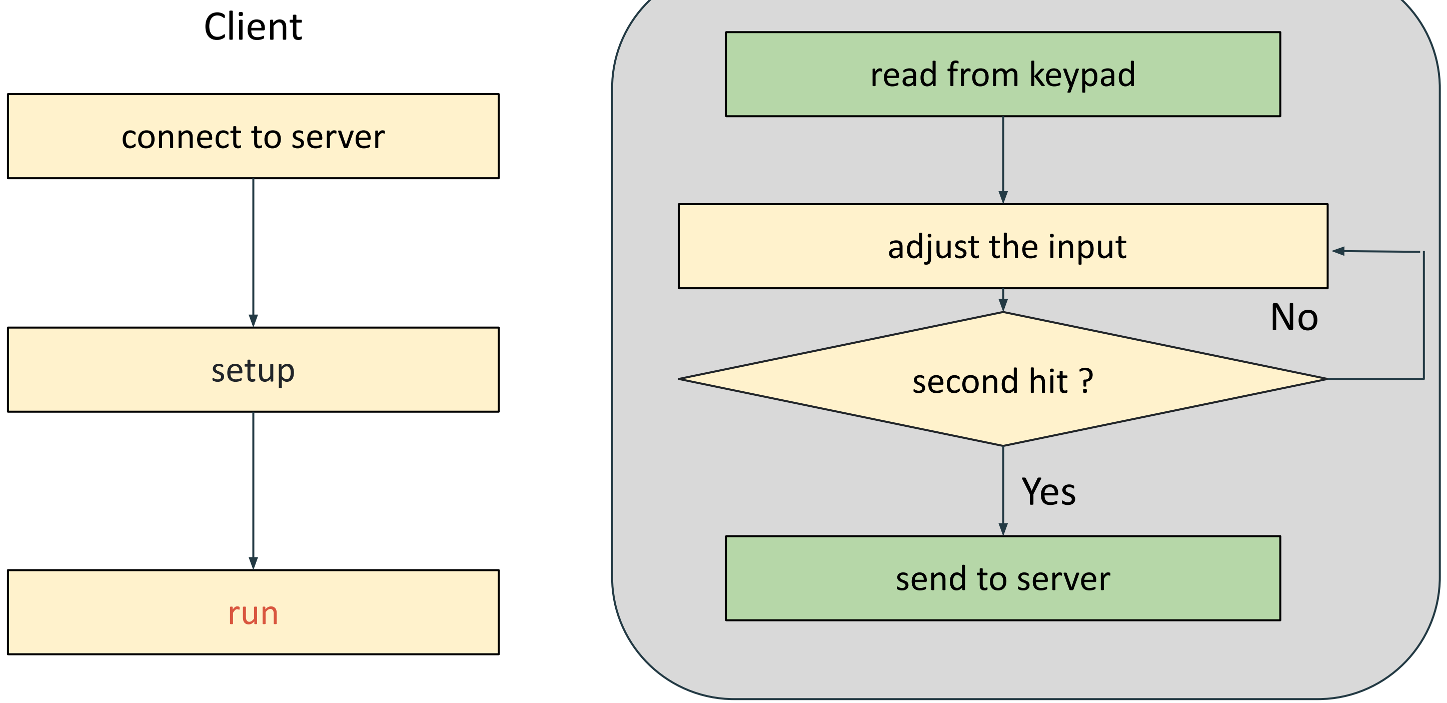
系統架構 - 流程圖



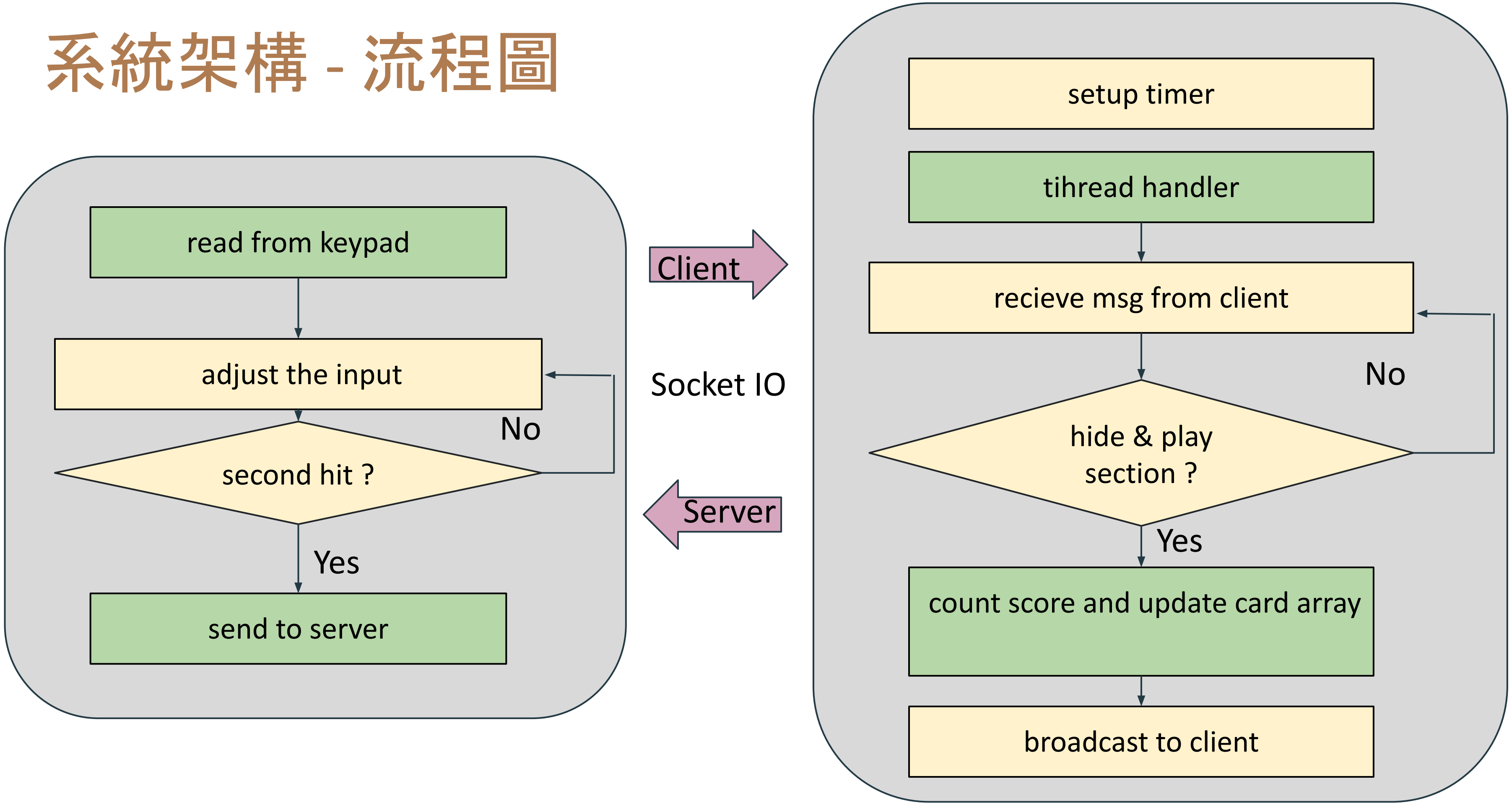


```
int main(int argc, char** argv)
{
    printf("Start game now\n");
    printf("Size of short = %d\n",sizeof(unsigned short));
    if(argc != 3)
        exit(printf("Usage: %s [IP address] [port]\n",argv[0]));
    if((socketfd = connectsock(argv[1],argv[2],"tcp")) < 0)
        exit(printf("Connect failed!!!\n"));

    if((io_fd = open("/dev/lcd", O_RDWR)) < 0)
        exit(printf("Open LCD module failed!!!!!\n"));

    ioctl(io_fd, LCD_IOCTL_CUR_OFF, NULL);
    ioctl(io_fd, LCD_IOCTL_CLEAR, NULL);
    ioctl(io_fd, KEY_IOCTL_CLEAR, key);
    //game.setup(0, io_fd);
    game.setup(socketfd, io_fd);

    pthread_create(&thread,NULL,server_Listener,NULL);
    printf("Servrer listener created\n");
    pthread_detach(thread);

    game.run();

}
```

系統架構 - 流程圖


Client
connect to server
setup
run
read from keypad
adjust the input
second hit ?
No
Yes
send to server

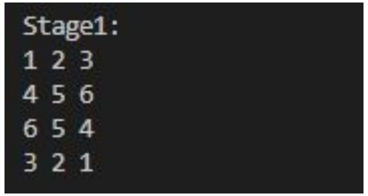
系統架構 - 流程圖



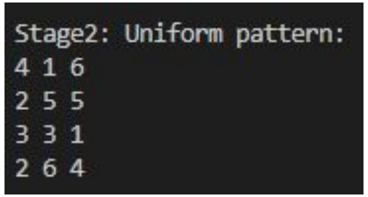
read from keypad
adjust the input
second hit ?
No
Yes
send to server
Client
Socket IO
Server
setup timer
tihread handler
recieve msg from client
hide & play section ?
No
Yes
count score and update card array
broadcast to client

隨機產生與圖形顯示

- 目標: 每回合隨機產生一組4x3大小的圖形陣列
- 方法: 使用1~6的數字對照6組圖案
- 數字陣列產生方法:
  1. 鏡像產生
  2. 隨機調換兩個數字N次

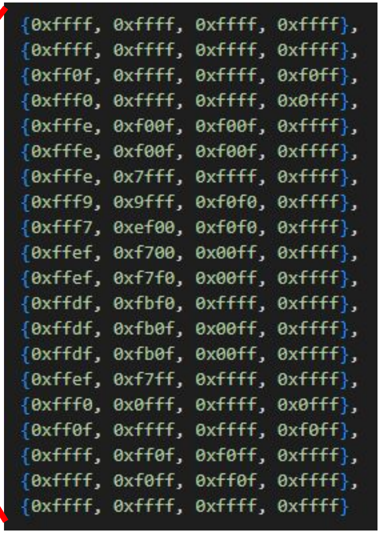


```
Stage1:
1 2 3
4 5 6
6 5 4
3 2 1
```



```
Stage2: Uniform pattern:
4 1 6
2 5 5
3 3 1
2 6 4
```

Mapping



```
{0xffff, 0xffff, 0xffff, 0xffff},
{0xffff, 0xffff, 0xffff, 0xffff},
{0xff0f, 0xffff, 0xffff, 0xf0ff},
{0xfff0, 0xffff, 0xffff, 0x0fff},
{0xfffe, 0xf00f, 0xf00f, 0xffff},
{0xfffe, 0xf00f, 0xf00f, 0xffff},
{0xfffe, 0x7fff, 0xffff, 0xffff},
{0xfff9, 0x9fff, 0xf0f0, 0xffff},
{0xfff7, 0xef00, 0xf0f0, 0xffff},
{0xffef, 0xf700, 0x00ff, 0xffff},
{0xffef, 0xf7f0, 0x00ff, 0xffff},
{0xffdf, 0xfbf0, 0xffff, 0xffff},
{0xffdf, 0xfb0f, 0x00ff, 0xffff},
{0xffdf, 0xfb0f, 0x00ff, 0xffff},
{0xffef, 0xf7ff, 0xffff, 0xffff},
{0xfff0, 0x0fff, 0xffff, 0x0fff},
{0xff0f, 0xffff, 0xffff, 0xf0ff},
{0xffff, 0xff0f, 0xf0ff, 0xffff},
{0xffff, 0xf0ff, 0xff0f, 0xffff},
{0xffff, 0xffff, 0xffff, 0xffff}
```

圖形顯示

- 顯示方式:
  1. 寫入全蓋牌畫面
  2. 根據Server傳過來的4x3陣列決定寫入哪些位置的牌
  3. 顯示在LCD上

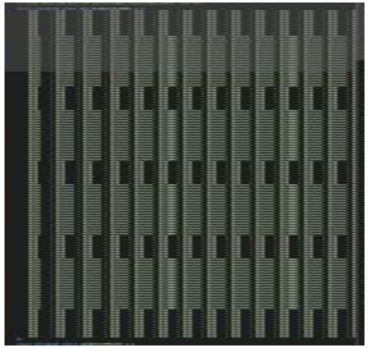


圖: 全蓋牌陣列

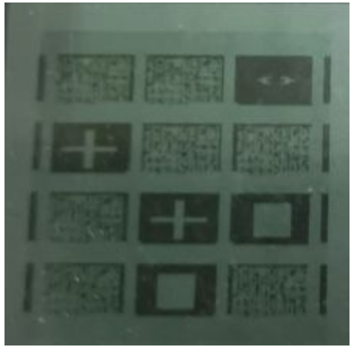
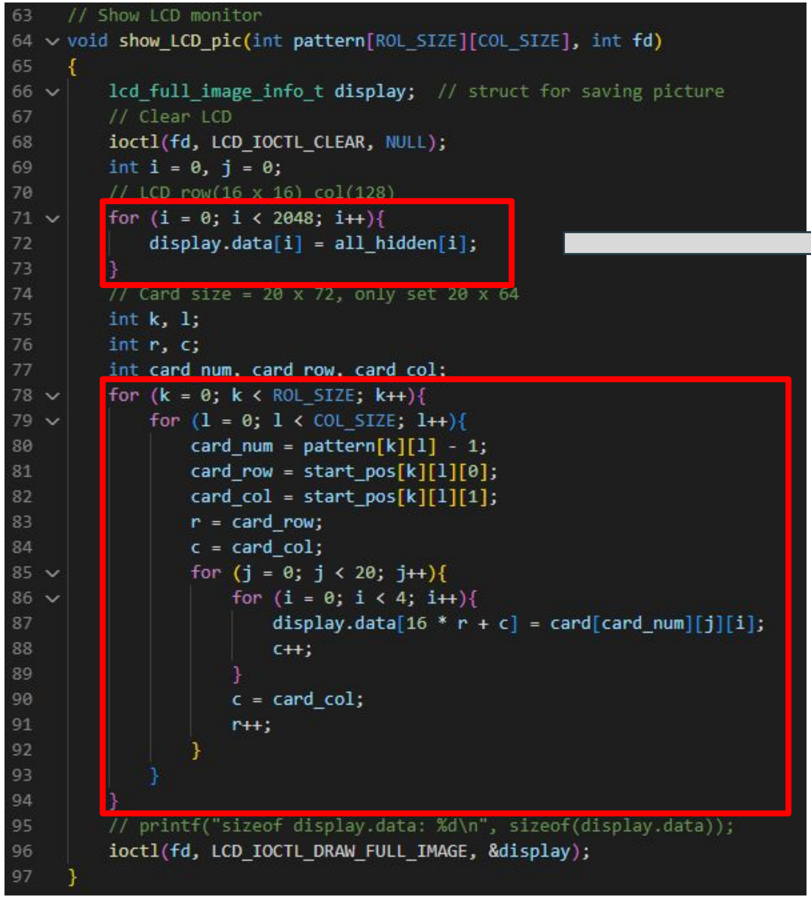


圖: 實際效果

此時Server傳來的陣列為

```
0 0 7
5 0 0
0 5 3
0 3 0
```

圖形顯示

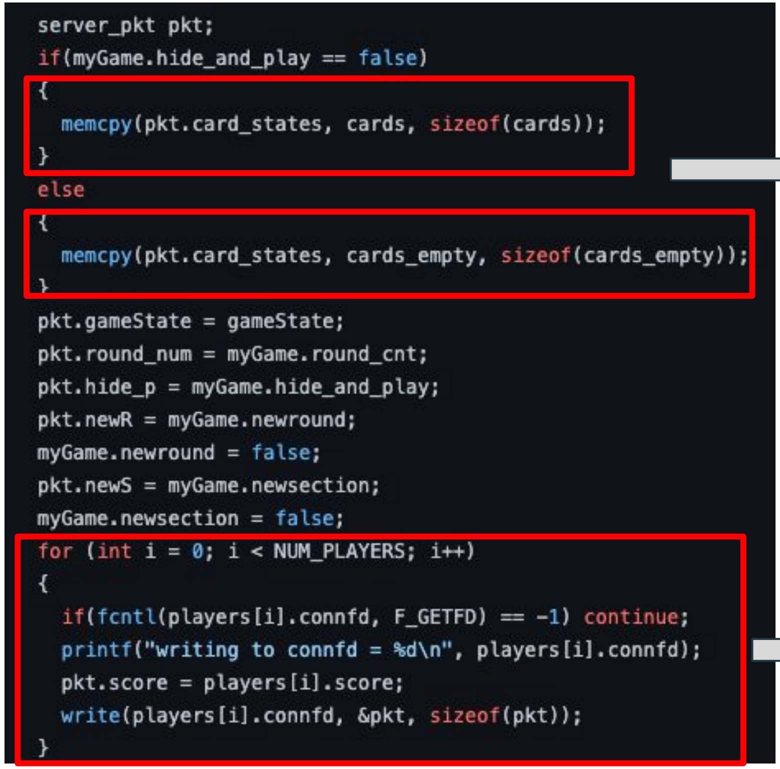


```c
// Show LCD monitor
void show_LCD_pic(int pattern[ROL_SIZE][COL_SIZE], int fd)
{
    lcd_full_image_info_t display;  // struct for saving picture
    // Clear LCD
    ioctl(fd, LCD_IOCTL_CLEAR, NULL);
    int i = 0, j = 0;
    // LCD row(16 x 16) col(128)
    for (i = 0; i < 2048; i++){
        display.data[i] = all_hidden[i];
    }
    // Card size = 20 x 72, only set 20 x 64
    int k, l;
    int r, c;
    int card_num, card_row, card_col;
    for (k = 0; k < ROL_SIZE; k++){
        for (l = 0; l < COL_SIZE; l++){
            card_num = pattern[k][l] - 1;
            card_row = start_pos[k][l][0];
            card_col = start_pos[k][l][1];
            r = card_row;
            c = card_col;
            for (j = 0; j < 20; j++){
                for (i = 0; i < 4; i++){
                    display.data[16 * r + c] = card[card_num][j][i];
                    c++;
                }
                c = card_col;
                r++;
            }
        }
    }
    // printf("sizeof display.data: %d\n", sizeof(display.data));
    ioctl(fd, LCD_IOCTL_DRAW_FULL_IMAGE, &display);
}
```

全蓋牌

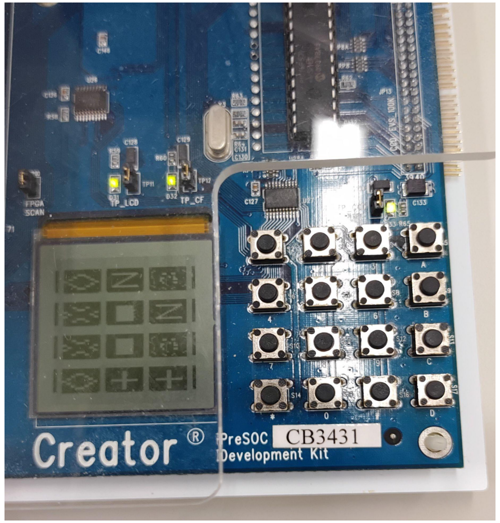
根據位置顯示其圖案

遊戲畫面 - Display Section



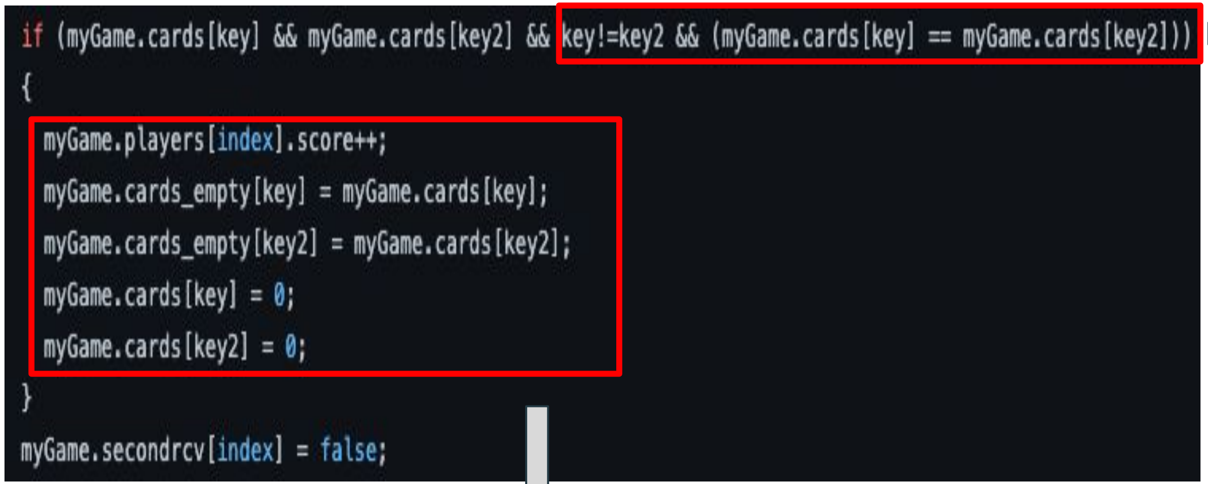
```
server_pkt pkt;
if(myGame.hide_and_play == false)
{
  memcpy(pkt.card_states, cards, sizeof(cards));
}
else
{
  memcpy(pkt.card_states, cards_empty, sizeof(cards_empty));
}
pkt.gameState = gameState;
pkt.round_num = myGame.round_cnt;
pkt.hide_p = myGame.hide_and_play;
pkt.newR = myGame.newround;
myGame.newround = false;
pkt.newS = myGame.newsection;
myGame.newsection = false;
for (int i = 0; i < NUM_PLAYERS; i++)
{
  if(fcntl(players[i].connfd, F_GETFD) == -1) continue;
  printf("writing to connfd = %d\n", players[i].connfd);
  pkt.score = players[i].score;
  write(players[i].connfd, &pkt, sizeof(pkt));
}
```

cards: 圖形陣列
cards_empty: 蓋牌陣列



發送分數, 圖案資訊給各個client

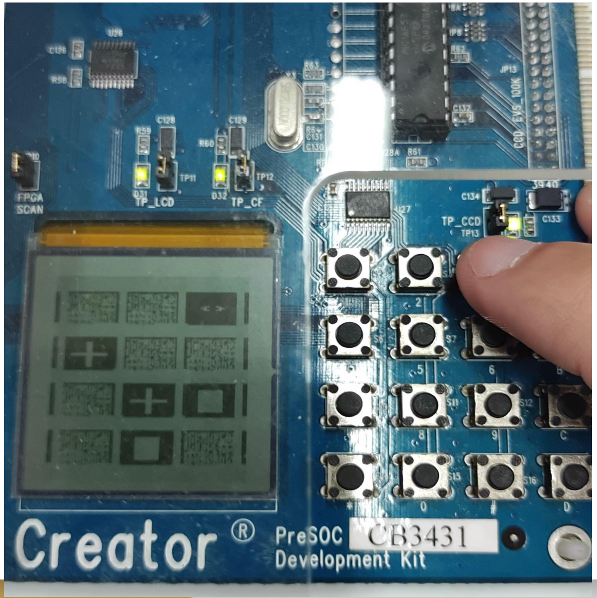
遊戲畫面 - Hide and play section



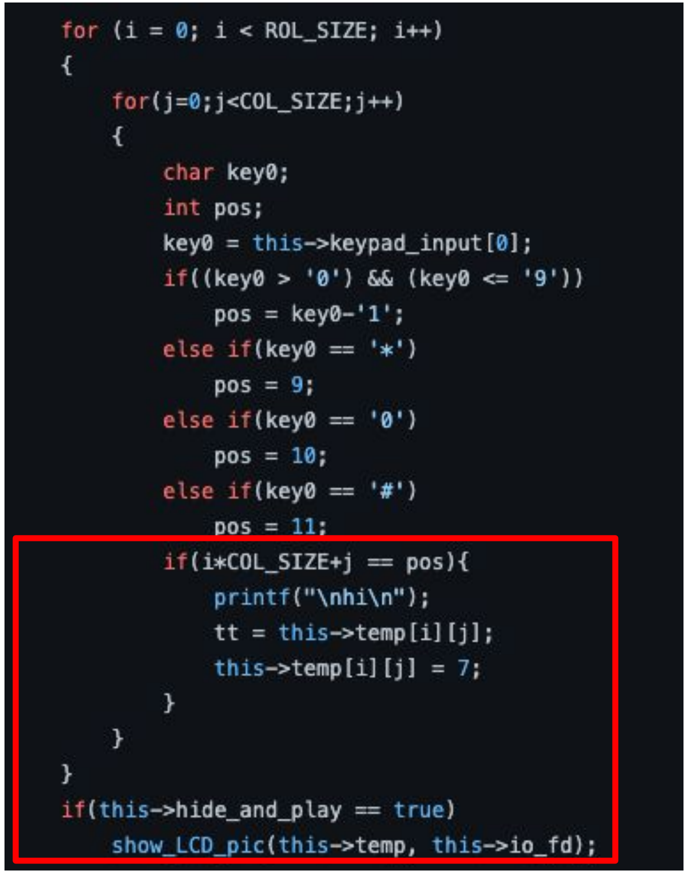
```
if (myGame.cards[key] && myGame.cards[key2] && key!=key2 && (myGame.cards[key] == myGame.cards[key2]))
{
  myGame.players[index].score++;
  myGame.cards_empty[key] = myGame.cards[key];
  myGame.cards_empty[key2] = myGame.cards[key2];
  myGame.cards[key] = 0;
  myGame.cards[key2] = 0;
}
myGame.secondrcv[index] = false;
```

判斷是否按到相同的圖案

在LCD面板上顯示正確配對的組合

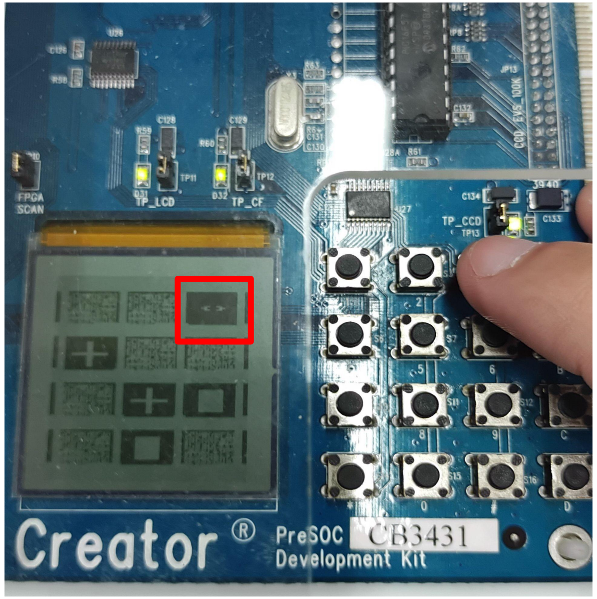


遊戲畫面 - Hide and play section



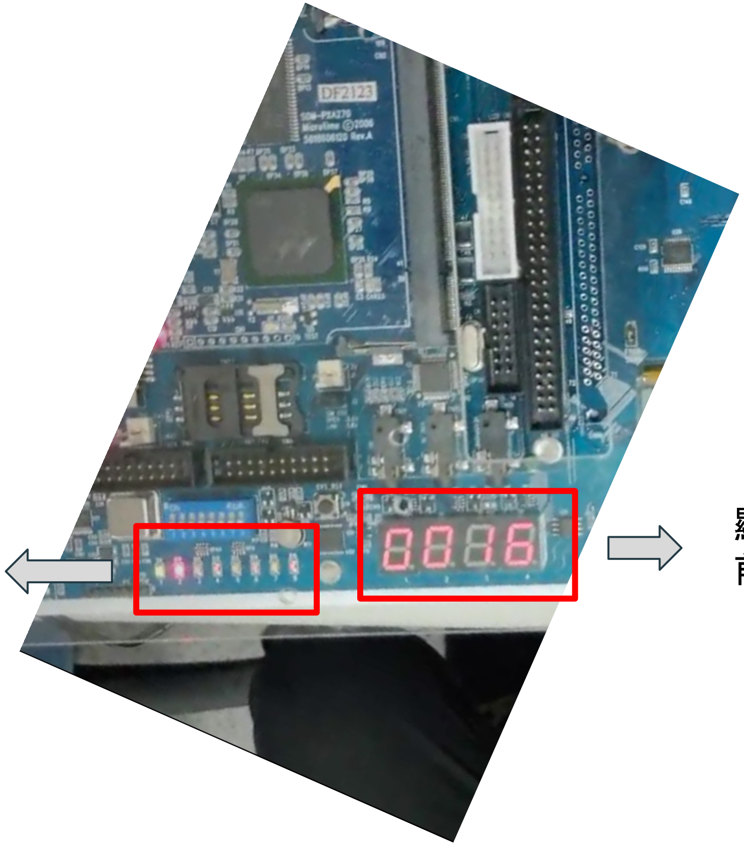
```
    for (i = 0; i < ROL_SIZE; i++)
    {
        for(j=0;j<COL_SIZE;j++)
        {
            char key0;
            int pos;
            key0 = this->keypad_input[0];
            if((key0 > '0') && (key0 <= '9'))
                pos = key0-'1';
            else if(key0 == '*')
                pos = 9;
            else if(key0 == '0')
                pos = 10;
            else if(key0 == '#')
                pos = 11;
            if(i*COL_SIZE+j == pos){
                printf("\nhi\n");
                tt = this->temp[i][j];
                this->temp[i][j] = 7;
            }
        }
    }
    if(this->hide_and_play == true)
        show_LCD_pic(this->temp, this->io_fd);
```

提示使用者
現在按了哪
一個隱藏牌



遊戲畫面 - 分數顯示

提示玩家剩
餘的回合數



顯示玩家目
前的分數

Demo影片

https://youtu.be/DSxPIYOiOls

團隊分工

309512074	黃柏叡	硬體建置、Game Client, Client Server整合測試
310512009	陳懿	硬體建置、Game Server, Client Server整合測試
310512025	賴知榆	除錯、隨機圖形產生器、LCD顯示

共同合作:
程式碼合併, 遊戲測試, 報告