Lab 4 Third Steps to Design

Leqi Ding
dlq991118@sjtu.edu.cn

Ming Cheng
chengming@sjtu.edu.cn

Wanda Guo
wdguo0424@gmail.com

August 12, 2019

Contents

Abstract

Welcome to the world of the Gomoku! In this lab we successfully implemented the Gobang game. When there are five pieces in the same row, column, or slash, the winner of the game appears. We used Xbee's wireless transmission function to implement information interaction between two arduino boards. When the winner appeared, we used lcd to display the prompt message. In addition, we also use a light sensor to control the rate at which the buzzer plays music.

Keywords: Gomuku, Xbee, wireless transmission, buzzer, light sensor

1 Introduction

In the final project, we design a game implementing two players playing Gomoku on PC, using the LCD display, serial monitor, two XBee boards, two Arduino boards and the technology of the communication between them. The checkerboard will be displayed on the Serial Monitor where the users will play Gomoku and the communication between two boards lets the users play in a very convenient way. Also, Happy Birthday will be played when a player wins, whose rhythm changes with the intensity of light, implementing with a light sensor. Besides, two different statements will be displayed on two LCD screens which indicates the win and lose of each player.

2 Discussion of the lab

2.1 Brief Design Specification

The whole project aims to continue to learn and practice the formal design process as we specify and develop a more complex system. We combine multiple functions to realize the Gomoku in wireless transmission and play music in the winner's buzzer. In addition, the rate of playing music can also be changed by changing the light intensity.

In our design of the Gomoku game, two players alternately enter the position of their pieces from the serial monitor. Before each input, the screen will output "It is your turn" and "wait!" to prompt the players. After entering the position of the piece, the serial monitors of the two ides will update the status of the board in real time. When there are five identical pieces in the same row or in the same column, the program will determine the winner of the game. "You win! Congratulations!" will be displayed on the winner's lcd, "You lose! Try again!" will be displayed on the loser's lcd. In addition, the winner's buzzer will play happy birthday music. In addition, with the light sensor, the rate at which the buzzer plays music varies with the intensity of the external light. The stronger the external light, the slower the playback rate. The weaker the light intensity, and the faster the playback rate.

2.2 Hardware Implementation

The connection of hardware is shown as the image below. In the project, we use two groups of Arduino boards, each of which contains a light sensor, a buzzer, an LCD Displayer, an Arduino UNO board, an Xbee and a Bus Shield. Two Xbees are used to implement the wireless connection. Buzzers are music players. Light sensors are speed controllers of playing music. Both board groups need to connect to the same PC.

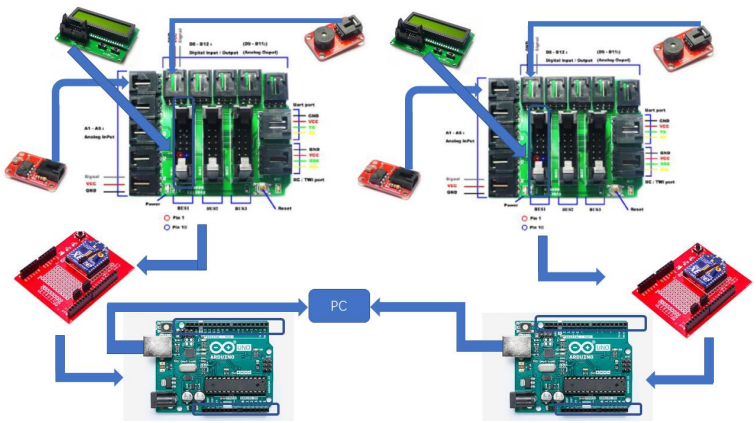

Figure 1: The diagram of hardware implementation

2.3 Software Implementation

To make our program easier to understand, in this part the program is divided to several parts. There are three main and important parts in our program. Two programs for player 1 and player 2 are similar to each other, but there are some minor differences, which will be mentioned in this part.

2.4 Initialization and Update of the Grid

To simplify the code in the main function, we design two functions, one to generate the gird, another to update the grid. In both functions, we use a bidimensional array to store the default symbol and symbols designed for two players. Then we print the symbols line by line. In the function for initialization, all the elements in the array is given the plus sign. In the function for update, the specific place is updated by the symbol which distinguishes two players. Hence, the codes for updating are for two players because the kind of their chess is different.

2.5 Music Player

In order to be able to play music with a buzzer, we convert the score of the song into a code form of the C language. We define Scale[ ] to store the music's notation, durt[ ] to store the duration of each note. Then we call the tone( ) function to let the buzzer play a specific frequency of sound, the duration of each sound depends on the value in durt[ ].

In addition, we have introduced a light sensor. When the external light intensity is large, the rate of playing music will be slower; when the external light intensity is small, the rate of playing music will be faster. To implement it, we connect the light sensor to port A5, then call the function analogRead( ) to read the analog value of the light sensor, then we change the duration of each note to the proportional function of the light intensity. So that we can achieve the function we expected.

2.6 Judgement for Winning Condition

To judge whether a player wins or not, we should code following the rules of Gomoku. In order to simplify the code and make best use of the memory, we only need to judge the situation after the latest piece is placed, regardless of the previous pieces. The variable cur records the latest piece placed. Also, the characters 'O' and 'X' represent two players.

According to the rules of Gomoku, a player wins only if five pieces are connected together in the vertical, horizonal or two diagonal directions. When checking in the vertical direction, we should think of the situations

that the current piece of chess will be placed upper or lower than the previous one, with different boundary situations. If the current piece is placed upper than the previous one, just as the following example code shows, the statement for(i = x-1,j = y; i >= 0 && count++ < num; i –) will be executed until i is less than zero or count is less than num. (We set num = 5 in the game of Gomoku and we can change its value if we want to play other chess games.)

Each time the for statement is executed, the variable winflag will plus one if a piece of chess is placed. So does other boundary situations. After the for statement ends being executed, a player will win if winflag is larger than num. In other situations like the judgement in horizonal direction or other directions, the for statements are similar to each other, except the boundary judgment.

2.7 The Main Program:

In the main program, we define a variable to judge whose turn it is to input the place of chess. If it is player one's turn, his serial monitor will print 'it's your turn to enter a number:'. If not, the words will be 'wait!'. The condition is same for player 2.

The function of handling invalid input is also implemented in the main program. In the loop function, to update the same thing in both serial monitors, we use an if statement to distinguish the input from two players and deny the invalid input in the meantime.

The last part of the code is to print words on LCD to inform both players of their playing results. If one player wins, the codes of winning works and the congratulations words will be printed on his LCD displayer. Meanwhile, the other part of the code does similar work on the other player's LCD and tells him he loses the game.

3 Test Plan

The first thing we need to verify is that when the player enters the position of the piece, both Arduino boards can update the chessboard synchronously. At the same time, we want to verify that the player 1's piece is displayed as 'X', the player 2's piece is displayed as 'O', and the position without piece is marked as '+'.

Secondly, we need to determine whether the program can judge the end of the game. When there are five identical pieces arranged in a row or a column, we think that the game wins. At this time, It will display "You Win! Congratulations!" on the winner's lcd, and display "You lose! Try again!" on the loser's lcd. At the same time, the winner's buzzer will play music.

Lastly, we need to verify that the light sensor we used is working. When the intensity of the light is large, the music played is relatively slow; when the intensity of the light is relatively small, the concert played is faster. We can use items to block the light sensor to observe the rate of music played.

4 Presentation, discussion and analysis of the results

Based upon the test plan, we tested several cases to check if our game works smoothly. The instructions of this game are in the user manual in the appendix of the report. We design five pairs of input by player 1 and player 2 to test the function of wireless transmission, condition of winning and solution of invalid input style such as placing a chess on a point where it has already been a chess. When opening the game, both of players gets a 15 by 15 grid printed. The same simple instructions are printed on the second line. The screenshot for the original state is shown below.

Figure 2: The chess board displayed on the screen

When one of the players get five chess on a straight line, he wins the game and the program stops running. Two LCD connected to the boards display the word shown below. As the words are displayed, the buzzer connected to the board of the winner starts playing Happy Birthday music repeatedly. The speed of music playing is controlled by the light intensity. When the surrounding gets darker, the buzzer plays the music faster, and vice versus.

5 Summary and Conclusion

In the final project, we successfully use two Arduino boards and two XBee boards to implement the communication between two users, through setting Micro and USB states of the XBee board, in order to let the users play a game - Gomoku.

The checkerboard is displayed on the Serial Monitor to let the users play on PC which will be updated every time a player plays, with 'O' and 'X' representing two players. After the game, there will be two different statements on two LCD screens indicating which one wins and which one loses. In the meanwhile, Happy Birthday will be played to celebrate the win of a player.

In order to implement the game as well as other functions, there are high requirements for hardware connections and software design. As for hardware connections, we need to connect the Arduino boards and XBee boards to the PC successfully and the correct setting of the states of the XBee boards is also very important. What's more, the connection between Arduino boards and the LCD should also be paid attention to. As for software design, the display of the checkerboard and the code to judge the game result are of great importance. In addition, we code to let the rhythm of the music (We choose Happy Birthday because of the limit of the memory.) be a function of the intensity of light, realizing the adjustable rhythm of the music.

Last but not least, we can implement the design of other chess games besides the Gomoku, like six-game chess just by changing a variable in the code, indicating the robustness and wide applicability of our code.

The communication between two Arduino boards is implemented by setting an appropriate state of the XBee board.

In this final project, we get much more familiar with the Arduino board, the LCD board, and the XBee board and we can also use them to implement the game design process - Gomoku. During these four labs, we improve the ability to design a game by ourselves and to solve problems by ourselves. Also, the ability to cooperate with others also gets great progress. As a recommendation of the project, we think it essential to have more introductive content of Arduino and correct some errors of instructions in the Lab Intro document.

6 Appendix

6.1 User Manual for Gomoku

Welcome to the world of Gomoku! Place your chess to get five chess pieces in a column, row or a slash! Meanwhile don't forget to block your opponent to prevent him from connecting five chess earlier than you! Bwahahahaha!

Rules of the Game

This is a two-player game. The players are named player 1 and player 2. Player 1 is prior to player 2 and place the chess pieces first. Then player 2 place the chess. Then player 1's turn. Then player 2's turn. One player wins if five chess pieces of him is in the same row or in the same column. Then the game ends. The 15 by 15 grid is displayed as 225 plus signs.

Operations of the Game

When the game starts, both Xbees are in USB mode. When it is turn for player 1, the Xbee connected to the board of program for player 1 is in USB mode and the Xbee for player 2 is in Micro Mode.

Player 1: You are given the priority to place your chess piece first. Before you input your decision, make sure that your Xbee is in USB mode and your opponent's Xbee is in Micro mode. Then input your decision of place of your chess piece into the serial monitor. Valid input should be a number from 0 to 224, which corresponds to the place on the grid. Any place on the grid can only place one chess piece in the game. Press enter to transmit the data into the program. Finally, change the mode of Xbee on your board to Micro mode and inform your opponent to change his Xbee to USB mode. Your chess pieces are displayed as crosses.

Player 2: You have to place your piece after player 1 finishes his turn. Before you input your decision, make sure that your Xbee is in USB mode and your opponent's Xbee is in Micro mode. Then input your decision of place of your chess piece into the serial monitor. Valid input should be a number from 0 to 224, which corresponds to the place on the grid. Any place on the grid can only place one chess piece in the game. Press enter to transmit the data into the program. Finally, change the mode of Xbee on your board to Micro mode and inform your opponent to change his Xbee to USB mode. Your chess pieces are displayed as circles. Winner judgement

The player who gets five chess pieces in the same row or in the same column without being blocked by his opponent, he wins the game. The winner will be awarded a happy birthday song played by his buzzer and congratulations words on LCD displayer. His opponent will be informed by 'try again' words on his LCD and no song is played on his buzzer.

Notes: The number of each places are shown below:

0,1,2…13,14(row 1)

15,16…,28,29(row 2)

…

210,211…223,224(row 15)

code

```
//player1
#include <LiquidCrystal.h>
#define ROW  15
#define COL  15
#define Do 262
#define Re 294
#define Mi 330
#define Fa 349
#define Sol 392
#define La 440
#define Si 494
#define Do_h 523
#define Re_h 587
#define Mi_h 659
#define Fa_h 698
#define Sol_h 784
#define La_h 880
#define Si_h 988
#define AD5 A5   //analog port

int intensity = 0;//intensity of light
int length;
int scale[]={Sol,Sol,La,Sol,Do_h,Si,
             Sol,Sol,La,Sol,Re_h,Do_h,
             Sol,Sol,Sol_h,Mi_h,Do_h,Si,La,
             Fa_h,Fa_h,Mi_h,Do_h,Re_h,Do_h};        //Happy birthday song
float durt[]=
{
  0.5,0.5,1,1,1,1+1,
  0.5,0.5,1,1,1,1+1,
   0.5,0.5,1,1,1,1,1,
    0.5,0.5,1,1,1,1+1,
};
    int buzzerPin = 12;           //the port of buzzer
    int BPM = 400;                //the unit time
    int Scale = 4;
    float rate = 0.1;

//const int ledPin = 12; // the pin that the LED is attached to
//int incomingByte;      // a variable to read incoming serial data into
char squareForGame[15][15] = {'0'};
int player = 1, i, choice;
char mark;
const int buzzer = 9;
char calSignal;
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7,d8=8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7,d8);
int x=0,y=0;
void Winner();

bool is_win(char squareForMap[15][15],int x,int y,int num);

void setup() {
```

```
  // initialize serial communication:
  Serial.begin(9600);
      lcd.begin(16, 2);
  //pinMode(buzzer, OUTPUT);// Set buzzer - pin 9 as an output
    pinMode(buzzerPin, OUTPUT);// Set buzzer - pin 9 as an output
    board();
    length=sizeof(scale)/sizeof(scale[0]);

   //Serial.print("Please enter x: ");
}

void loop() {
  if(player % 2 == 1)
    Serial.print("it's your turn to enter a number: ");
  else
    Serial.print("wait!");
  while(true)
  {
  if(Serial.available()>0)
  {
    x = Serial.parseInt();
    Serial.println(x);

   if(player % 2 == 0 && squareForGame[(x) % 15][(x) / 15]== '+')
    {squareForGame[(x) % 15][(x) / 15]='X';}
    else if(player % 2 == 1 && squareForGame[(x) % 15][(x) / 15]== '+')
    {squareForGame[(x) % 15][(x) / 15]='O';}
    else
    {
      Serial.print("Invalid input!");
      player--;
    }
    if(is_win(squareForGame,(x) % 15,(x) / 15,5) && player % 2 == 1)
    {
      lcd.setCursor(0,0);
      lcd.print("You win!");
      lcd.setCursor(0,1);
      lcd.print("Congratulations!");
      while(true)
      {
        Winner();
      }

    }
    else if(is_win(squareForGame,(x) % 15,(x) / 15,5) && player % 2 == 0)
    {
      lcd.setCursor(0,0);
      lcd.print("You lose! ");
      lcd.setCursor(0,1);
      lcd.print("Try again!");
       while(1){}
    }
    boardupdate();
   player++;
   break;
  }
  }
}
```

```
void board()
{
    Serial.print("\n\n\tGomoku\n\n");

    Serial.print("Player 1 (X)  -  Player 2 (O)\n\n\n");

    int a = 0;
   // Serial.print("|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |\n");
    while(a < 15)
    {
      Serial.print("  ");
      Serial.print(squareForGame[0][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[1][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[2][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[3][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[4][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[5][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[6][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[7][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[8][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[9][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[10][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[11][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[12][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[13][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[14][a]='+');
      Serial.print("    ");
      Serial.print(" \n");
      Serial.print(" \n");

 //Serial.print("|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|__
___|_____|_____|_____|_____|\n");
      a++;
    }

      //Serial.print("|     |     |     |     |     |     |     |     |     |
  |     |     |     |     |     |\n");
}
void boardupdate()
{
    int a = 0;
   // Serial.print("|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |\n");
```

```
    while(a < 15)
    {
      Serial.print("  ");
      Serial.print(squareForGame[0][a]);
      Serial.print("    ");
      Serial.print(squareForGame[1][a]);
      Serial.print("    ");
      Serial.print(squareForGame[2][a]);
      Serial.print("    ");
      Serial.print(squareForGame[3][a]);
      Serial.print("    ");
      Serial.print(squareForGame[4][a]);
      Serial.print("    ");
      Serial.print(squareForGame[5][a]);
      Serial.print("    ");
      Serial.print(squareForGame[6][a]);
      Serial.print("    ");
      Serial.print(squareForGame[7][a]);
      Serial.print("    ");
      Serial.print(squareForGame[8][a]);
      Serial.print("    ");
      Serial.print(squareForGame[9][a]);
      Serial.print("    ");
      Serial.print(squareForGame[10][a]);
      Serial.print("    ");
      Serial.print(squareForGame[11][a]);
      Serial.print("    ");
      Serial.print(squareForGame[12][a]);
      Serial.print("    ");
      Serial.print(squareForGame[13][a]);
      Serial.print("    ");
      Serial.print(squareForGame[14][a]);
      Serial.print("    ");
      Serial.print(" \n");
      Serial.print(" \n");

 //Serial.print("|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|\n");
      a++;
    }
}


bool is_win(char squareForMap[][],int x,int y,int num)
{


    int count = 0;
    int winflag = 1;// The first point does not need to be read again
    char cur ;// record the current chess
    int i,j;
    // 'O' and 'X' represent two pieces
    cur = squareForMap[x][y] == 'O' ? 'O' : 'X';

    //judge in vertical direction

    //up
```

```
    for(i = x-1,j = y; i >= 0 && count++ < num; i --) // count num times or stop
at the checkerboard boundary
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break; // not the same piece
    }
    //down
    count = 0;//clear the count
    for(i = x + 1,j = y; i < ROW && count ++ < num; i ++ )
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;//clear
    if( winflag >= num )// end reading in the vertical direction
        return true;
    else
        winflag = 1;

    //judge in horizontal direction
    //right
    for(i = x,j = y + 1; j < COL && count ++ < num ; j ++ )
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
    //left
    for(i = x, j = y - 1; j >= 0 && count++ < num ; j --)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0; // clear
    // end reading in the horizontal direction
    if( winflag >= num )
        return true;
    else
        winflag = 1;

    //judge in the first diagonal direction
    //right lower direction
    for(i = x + 1,j = y + 1; i < ROW && j < COL && count ++ < num ; i ++ ,j ++)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
```

```
    //Upper left direction
    for(i = x - 1,j = y - 1; i >= 0 && j >= 0 && count++ < num ; i -- ,j --)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
    //end reading in the first diagonal direction
    if( winflag >= num )
        return true;
    else
        winflag = 1;

    //judge in the second diagonal direction
    //Upper right direction
    for(i = x -1,j = y + 1; i >= 0 && j < COL && count++ < num; i --,j++)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
    //Lower left direction
    for(i = x + 1,j = y -1; i < ROW && y >= 0 && count++ < num; i ++,j--)
    for(i = x + 1,j = y -1; i < ROW && y >= 0 && count++ < num; i ++,j--)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
    //end reading in the second diagonal direction
    if( winflag >= num )
        return true;
    else
        winflag = 1;

    //end reading in all directions
    return false;
}

void Winner()
{
      for(int x=0;x<length;x++)
  {
    tone(buzzerPin,scale[x]);
    intensity = analogRead(AD5);
    intensity *=1.524;
    delay(intensity*durt[x]);
    noTone(buzzerPin);
  }

  delay(2000);
}
```

```
//player2
#include <LiquidCrystal.h>
#define ROW  15
#define COL  15
#define Do 262
#define Re 294
#define Mi 330
#define Fa 349
#define Sol 392
#define La 440
#define Si 494
#define Do_h 523
#define Re_h 587
#define Mi_h 659
#define Fa_h 698
#define Sol_h 784
#define La_h 880
#define Si_h 988
#define AD5 A5   //define the port for the light sensor
#define LED 11

int intensity = 0;//light intensity
int length;
int scale[]={Sol,Sol,La,Sol,Do_h,Si,
             Sol,Sol,La,Sol,Re_h,Do_h,
             Sol,Sol,Sol_h,Mi_h,Do_h,Si,La,
             Fa_h,Fa_h,Mi_h,Do_h,Re_h,Do_h};        //Happy birthday song
float durt[]=
{
  0.5,0.5,1,1,1,1+1,
  0.5,0.5,1,1,1,1+1,
   0.5,0.5,1,1,1,1,1,
    0.5,0.5,1,1,1,1+1,
};
    int buzzerPin = 12;             //port of the buzzer
    int BPM = 400;                  //unit time
    int Scale = 4;
    float rate = 0.1;

//const int ledPin = 12; // the pin that the LED is attached to
//int incomingByte;      // a variable to read incoming serial data into
char squareForGame[15][15] = {'0'};
int player = 1, i, choice;
char mark;
const int buzzer = 9;
char calSignal;
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7,d8=8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7,d8);
int x=0,y=0;
void Winner();

bool is_win(char squareForMap[15][15],int x,int y,int num);

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
```

```
      lcd.begin(16, 2);
    pinMode(buzzerPin, OUTPUT);// Set buzzer - pin 9 as an output
    board();
    length=sizeof(scale)/sizeof(scale[0]);

   //Serial.print("Please enter x: ");
}

void loop() {
  if(player % 2 == 0)
    Serial.print("it's your turn to enter a number: ");
  else
    Serial.print("wait!");
  while(true)
  {
  if(Serial.available()>0)
  {
    x = Serial.parseInt();
    Serial.println(x);
    if(player % 2 == 0 && squareForGame[(x) % 15][(x) / 15]== '+')
    {squareForGame[(x) % 15][(x) / 15]='X';}
    else if(player % 2 == 1 && squareForGame[(x) % 15][(x) / 15]== '+')
    {squareForGame[(x) % 15][(x) / 15]='O';}
    else
    {
      Serial.print("Invalid input!");
      player--;
    }
    if(is_win(squareForGame,(x) % 15,(x) / 15,5) && player % 2 == 0)
    {
      lcd.setCursor(0,0);
      lcd.print("You win!");
      lcd.setCursor(0,1);
      lcd.print("Congratulations!");
      while(true)
      {
        Winner();
      }

    }
    else if(is_win(squareForGame,(x) % 15,(x) / 15,5) && player % 2 == 1)
    {
      lcd.setCursor(0,0);
      lcd.print("You lose!");
      lcd.setCursor(0,1);
      lcd.print("Try again!");
      while(1){}
    }
    boardupdate();
   player++;
   break;
  }
  }
}
void board()
{
    Serial.print("\n\n\tFIVE CHESS\n\n");
```

```
    Serial.print("Player 1 (X)  -  Player 2 (O)\n\n\n");

    int a = 0;
   // Serial.print("|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |\n");
    while(a < 15)
    {
      Serial.print("  ");
      Serial.print(squareForGame[0][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[1][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[2][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[3][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[4][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[5][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[6][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[7][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[8][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[9][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[10][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[11][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[12][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[13][a]='+');
      Serial.print("    ");
      Serial.print(squareForGame[14][a]='+');
      Serial.print("    ");
      Serial.print(" \n");
      Serial.print(" \n");

 //Serial.print("|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|__
___|_____|_____|_____|_____|\n");
      a++;
    }

      //Serial.print("|     |     |     |     |     |     |     |     |     |
  |     |     |     |     |     |\n");
}
void boardupdate()
{
    int a = 0;
   // Serial.print("|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |\n");
    while(a < 15)
    {
      Serial.print("  ");
      Serial.print(squareForGame[0][a]);
```

```
      Serial.print("    ");
      Serial.print(squareForGame[1][a]);
      Serial.print("    ");
      Serial.print(squareForGame[2][a]);
      Serial.print("    ");
      Serial.print(squareForGame[3][a]);
      Serial.print("    ");
      Serial.print(squareForGame[4][a]);
      Serial.print("    ");
      Serial.print(squareForGame[5][a]);
      Serial.print("    ");
      Serial.print(squareForGame[6][a]);
      Serial.print("    ");
      Serial.print(squareForGame[7][a]);
      Serial.print("    ");
      Serial.print(squareForGame[8][a]);
      Serial.print("    ");
      Serial.print(squareForGame[9][a]);
      Serial.print("    ");
      Serial.print(squareForGame[10][a]);
      Serial.print("    ");
      Serial.print(squareForGame[11][a]);
      Serial.print("    ");
      Serial.print(squareForGame[12][a]);
      Serial.print("    ");
      Serial.print(squareForGame[13][a]);
      Serial.print("    ");
      Serial.print(squareForGame[14][a]);
      Serial.print("    ");
      Serial.print(" \n");
      Serial.print(" \n");

 //Serial.print("|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|\n");
      a++;
    }
}


bool is_win(char squareForMap[][],int x,int y,int num)
{


    int count = 0;
    int winflag = 1;// The first point does not need to be read again
    char cur ;// record the current chess
    int i,j;
    // 'O' and 'X' represent two pieces
    cur = squareForMap[x][y] == 'O' ? 'O' : 'X';

    //judge in vertical direction

    //up
    for(i = x-1,j = y; i >= 0 && count++ < num; i --) // count num times or stop at the checkerboard boundary
    {
        if(squareForMap[i][j] == cur)
            winflag++;
```

```
        else
            break; // not the same piece
    }
    //down
    count = 0;//clear the count
    for(i = x + 1,j = y; i < ROW && count ++ < num; i ++ )
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;//clear
    if( winflag >= num )// end reading in the vertical direction
        return true;
    else
        winflag = 1;

    //judge in horizontal direction
    //right
    for(i = x,j = y + 1; j < COL && count ++ < num ; j ++ )
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
    //left
    for(i = x, j = y - 1; j >= 0 && count++ < num ; j --)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0; // clear
    // end reading in the horizontal direction
    if( winflag >= num )
        return true;
    else
        winflag = 1;

    //judge in the first diagonal direction
    //right lower direction
    for(i = x + 1,j = y + 1; i < ROW && j < COL && count ++ < num ; i ++ ,j ++)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
    //Upper left direction
    for(i = x - 1,j = y - 1; i >= 0 && j >= 0 && count++ < num ; i -- ,j --)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
```

```
        else
            break;// not the same piece
    }
    count = 0;
    //end reading in the first diagonal direction
    if( winflag >= num )
        return true;
    else
        winflag = 1;

    //judge in the second diagonal direction
    //Upper right direction
    for(i = x -1,j = y + 1; i >= 0 && j < COL && count++ < num; i --,j++)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
    //Lower left direction
    for(i = x + 1,j = y -1; i < ROW && y >= 0 && count++ < num; i ++,j--)
    for(i = x + 1,j = y -1; i < ROW && y >= 0 && count++ < num; i ++,j--)
    {
        if(squareForMap[i][j] == cur)
            winflag++;
        else
            break;// not the same piece
    }
    count = 0;
    //end reading in the second diagonal direction
    if( winflag >= num )
        return true;
    else
        winflag = 1;

    //end reading in all directions
    return false;
}

void Winner()
{
      for(int x=0;x<length;x++)
  {
    tone(buzzerPin,scale[x]);
    intensity = analogRead(AD5);
    intensity *=1.524;
    delay(intensity*durt[x]);
    noTone(buzzerPin);
  }

  delay(2000);
}
```