Lab 3 Second Steps to Design

Leqi Ding
dlq991118@sjtu.edu.cn

Ming Cheng
chengming@sjtu.edu.cn

Wanda Guo
wdguo0424@gmail.com

August 1, 2019

Contents

Abstract

Work to have fun, so does this lab. We design a game - a variation on the game Minesweeper in this lab and there are single-player version and two-player version of it. The goal of the game is to start from one of the four corners and get to the diagonally opposite corner avoiding the four mines which are set randomly in the minefield. Mines are set randomly in single-player version while in two-player version, however, mines are set by the other player. Also, there is a time requirement in the game. To play this game, there should be interaction between PC and Arduino boards, implementing using the Serial Monitor as the interface of the game.

Keywords: game, single-player, two-player, Arduino board, Serial Monitor

1 Introduction

This lab has two parts, both of which require to design a game - a variation on the game Minesweeper. The goal of the game is to start from one of the four corners and get to the diagonally opposite corner avoiding the four mines which are set randomly in the minefield. The first part is a single-player version while the second one is a two-player version, implemented by the interaction between two Arduino boards and two PCs. For the single-player version, the minefield will be displayed on the Serial Monitor and the move of the player is controlled by the keyboard. For the two-player version, however, each player will set four mines for the other player, utilizing a Arduino board connected to PC. The move of each player is controlled on two keyboards, respectively. Also, there is a time requirement in the game.

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 - design a game, both the single-player version and the two-player version. The map will be displayed on the Serial Monitor and the move of the player is controlled by the keyboard.

2.1.1 Phase1

In this phase, the game is in a single-player version. There are two lives for the player and the life will decrease if the player meets a mine. The locations of four mines are set randomly and the player can control the move by the ’w’, ’a’, ’s’ and ’d’ on the keyboard, which means moving to up, left, down and right, correspondingly. The map will be updated and displayed every time a command is inputted. The lives remained will be displayed all the time and there are some prompts when the game is over or the player wins.

2.1.2 Phase2

This phase is based on the previous one, implementing a two-player version game. Two Arduino boards are connected to two PCs respectively and there are also interactions between them. The locations of mines on each map are set by the other player and the moves of the players are also controlled by ’w’, ’a’, ’s’ and ’d’ on the keyboard. If both two players do not achieve the destination successfully, the first person who died will lose the game. And if both players achieve the destination successfully, the time cost of two players will determine which player wins - the player with less time cost may win, obviously.

2.2 Hardware Implementation

The schematic diagram of the hardware implementation is as above. In single-player version, we only need one board. The board is connected to PC and no other hardware connection is needed. In two-player version, we need two boards and the communication between two boards requires that both boards need to be connected to Bus Shield and port A4 and A5 of both boards are connected correspondingly. Both boards need to be connected to one PC to implement two ports which enable PC to open two serial monitors.

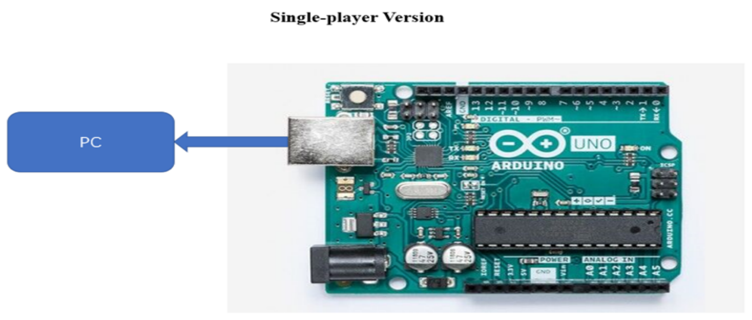

Figure 1: The schematic diagram of the hardware implementation in single-player game

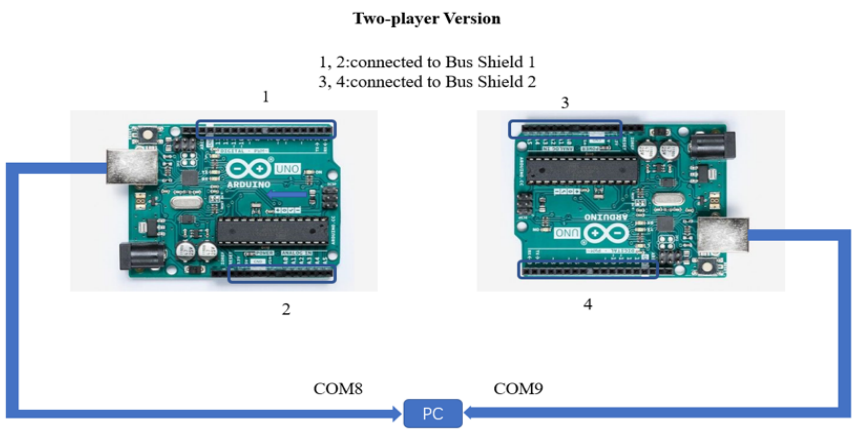
Two-player Version
1, 2:connected to Bus Shield 1
3, 4:connected to Bus Shield 2
1
3
2
4
COM8
COM9
PC

Figure 2: The schematic diagram of the hardware implementation in two-player game

2.3 Software Implementation

Experiment 3 is divided into two parts: the design of single player and two-player games. Their ideas on software implementation are as follows:

Single player game: First we consider how the data is stored. Since the entire area of the problem is a 4×4 grid, we used a two-dimensional array to store the data in the grid structure. Then we need to consider how the results are displayed. We use 'Y' to indicate the current position of the player, and the remaining positions are represented by '+'.

Then we need to randomly generate the location of the mine. We use x = rand()%4; y = rand()%4; to generate the horizontal and vertical coordinates of the mine position. We define another 4×4 two-dimensional array of mines to store the location of the mine. If there is a mine in a certain location, its value in the corresponding position in mines[][] is 1, otherwise it is 0.

In addition, we also need to achieve the interaction between the computer and the player. We have selected 'w', 'a', 's' and 'd' as the instructions for the move, specifically

w - move up

a - move left

s - move down

d - move right

We can get the information of the moving direction by comparing the four moving instructions with the data read from the serial monitor. In addition to this, we stipulate that if the current position is on the first line, when the instruction to move up is input, the next position will remain in place. In order to update the current location in real time, we define the xnow, ynow, and xlast, ylast four variables to store the position of the player before and after the move. This way we can update the player's current location in real time.

In order to increase the difficulty of the game, we also added a timing function. Based on the references we found, we introduced a new library. Timed by setting the interrupt mode, MsTimer2::set(30000, flash); MsTimer2::start(); We limit the time to 30 seconds. When the counter value reaches 30, the program will enter the interrupt service routine on the screen. The output 'Time is over! Try again!'.

Two-player game is based on the single player game. We need to add data interaction between the two arduino boards: we use tran_e = Serial.parseInt(); to enter the data that needs to be transferred, then the function requestEvent() will send the values of tran_e, tran_b, tran_c, tran_d to another arduino board which is waiting for input data. In addition, we also designed the data read function receiveEvent();, which realizes reading the sent data and setting the position of minefield. In order to facilitate the player to input data, we will prompt the player to input an integer from 1 to 14 to represent the position of mines.

Figure 3 shows the structural diagram on software.

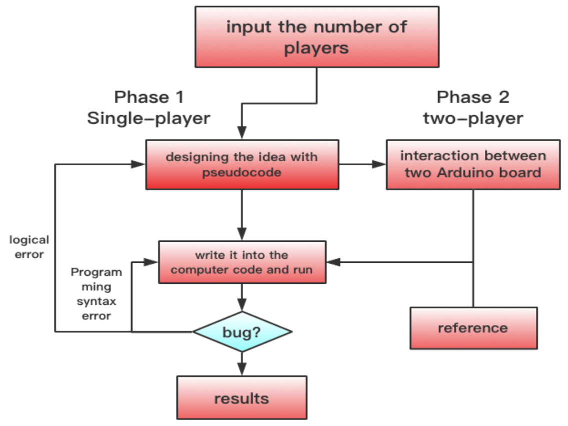
input the number of players
Phase 1
Single-player
Phase 2
two-player
designing the idea with pseudocode
interaction between two Arduino board
logical error
write it into the computer code and run
Program ming syntax error
bug?
reference
results

Figure 3: The structural diagram on software

3 Test Plan

In phase 1, we need to ensure the 4 mines placed randomly, the judgement of victory and loss and the input for shifting is effective.

In phase 2, we need to verify whether the selection of 2 groups of mines are properly sent to the other player's PC via two Arduino UNO boards and the basic function in phase 1 also works in phase 2.

4 Presentation, discussion and analysis of the results

The user manual presents the instructions of the Mines game we implemented.

In single-player version:

We run the program. The directions are printed. Then the player can enter the characters 'w', 'a', 's', 'd' to move in the grid. When the player input the wrong character that may move beyond the border, the place of the player won't be changed to keep the game running normally. When the player successfully go to the right bottom corner, the player wins. If player lose both of his lives, he loses the game. All the results are satisfactory.

In two-player version:

We run the program. Player 1 selects the place of mines in his opponent's grid first. Then player 2 selects the mines. Now, both players can start the game. The remaining procedure is completely the same as the single-player mode. We tested and checked whether the places of mines were true. Finally, we got the satisfactory result. Two players can play the Mines game without bugs.

5 Analysis of any errors

In phase 1, we encountered two problems. The first one is the form of display of the interface. After discussing the problem with TAs, we decided to print the game interface in serial monitor, and the board serves as an execution processor. The second problem we met is the mines are not randomly placed. We solved the problem by setting time seeds to ensure the randomness.

In phase 2, we met two problem that we find it difficult to handle. The first one is how to design the game to make two players select the places of mines in opponent's grid. Our solution is to make player 1 prior to player 2 and let player 1 enter first. Player 2 cannot select the mines until the selection of player 1 is transmitted to the serial port of player 2. Then after player 2 finished selecting, the data is transmitted to the same serial port. The problem is that we need to implement the transmission of data between two PCs via two boards. This method utilizes the I2C protocol and implement. We referred to the source code we wrote and successfully realize the function we need.

6 Summary and Conclusion

In this project, we successfully achieve utilizing Arduino boards and Serial Monitor to design a game - a variation on the game Minesweeper. The project consists of two phases, with the first one is a single-player version game while the second one is a two-player version game.

In the first one, we use four characters on the keyboard ('w', 'a', 's' and 'd') to control the moving directions and try to achieve the diagonally opposite corner as possible. The map of the game and other prompts will be displayed on the Serial Monitor. What's more, the time cost is also calculated when playing the game.

In the second one, two players use two different computers to set mines for each other and play games as well. The two computers are connected with two Arduino boards through which the interaction between them is implemented. The moving control is similar to the previous phase, with 'w', 'a', 's' and 'd' correspond to up, left, down and right. If both two players do not achieve the destination successfully, the first person who died will lose the game. On the contrary, if both players achieve the destination successfully, which one will win depends on the time cost of each other. What is also similar to the previous phase is that the map of the game will be displayed on the Serial Monitor and some prompts will be displayed at the end of the game as well.

Two boards are programmed to communicate with each other via the I2C synchronous serial protocol. Several functions of Arduino's Wire Library are used to accomplish this.

The I2C protocol involves using two lines to send and receive data: a serial clock pin (SCL) that the Arduino or Genuino Master board pulses at a regular interval, and a serial data pin (SDA) over which data is sent between the two devices. As the clock line changes from low to high (known as the rising edge of the clock pulse), a single bit of information - that will form in sequence the address of a specific device and a command or data - is transferred from the board to the I2C device over the SDA line. When this information is sent - bit after bit -, the called upon device executes the request and transmits it's data back - if required - to the board over the same line using the clock signal still generated by the Master on SCL as timing.

In this project, we get more familiar with the Arduino board and use it to implement the formal design process. 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.

7 Appendix

7.1 The instruction on the rules of our game

This game is derived from the original game called Mines. The goal of this game is to cross the minefield and 'survive'. The minefield is a 4 by 4 grid and there are 4 mines placed in the grid. You have 2 lives and if you are wounded by a mine for the second time, game is over and you should restart the game or you can quit the game by closing the window.

Operating Instructions

Phase 1 Single-player Version

The system generates 4 mines randomly and secretly. You start your game from the top left corner. You can't cross the four boundaries around the grid. The character 'Y' displays your place.

Input the characters below to move (one character at a time):

w-move up
a-move left
s-move down
d-Move right

If you can get to the bottom right corner of the grid and still have at least one life, you win. Or if you lose all of your lives, you lose the game and need to restart the game.

Phase 2 Two-player Version

In two-player version, you need to select the place of four mines in your opponent's grid. If you are player 1, you have the priority to select the places of mines in your opponent's grid. Then you can input the characters below to move and If you are player 2, you need to wait for player 1 to enter the places of mines in your grid. Then both player 1 and player 2 can start the game. Players enter the characters below to move:

w-move up
a-move left
s-move down
d-Move right

If you can get to the bottom right corner of the grid and still have at least one life, you win. Or if you lose all of your lives, you lose the game and need to restart the game.

```
//player1

#include <MsTimer2.h>
#include <Wire.h>

int hisxposition;
int hisyposition;
int myxposition;
int myyposition;
int hispunction;

int e = 0;
int b = 0;
int c = 0;
int d = 0;
int xnow = 0;
int ynow = 0;
int xlast = 0;
int ylast = 0;

int x;
int y;

int incomingByte = 0;
int numofplayer = 0;
int numofmines = 0;
int yourlifes = 2;
char Mines[4][4];
int mines[4][4] = {0};
int flag = 0;
int i=0;
int start=0;
int data[50];
int datain[50];
int symbol=0;
int symbol2=0;
int start2=0;

int tran_e = 0;
 int tran_b = 0;
 int tran_c = 0;
 int tran_d = 0;
 int firstTime =true;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Wire.begin(4);
  Serial.print("Please insert number of the players:");

  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);
/*
  Serial.println("The map: ");
  Serial.println("Y+++");
  Serial.println("++++");
  Serial.println("++++");
  Serial.println("++++");
  Serial.println('Your have 2 lifes now');
*/
  for(int i=0;i<4;i++)//initialize the array
     for(int j=0;j<4;j++)
     {
      if(i==0&&j==0)
      Mines[i][j] = 'Y';
      else
      {
        Mines[i][j] = '+';
      }

     }
  }

void loop() {
  // put your main code here, to run repeatedly:
  if(flag==1)
  {
    while(1)
    {

    }
  }
  if(yourlifes == 0)
  {
    Serial.println(' ');
    Serial.println("You are over.Hhhhh");
    while(1)
    {

    }
  }

  if(xnow==3&&ynow==3)
  {
    Serial.println(' ');
    Serial.println("You win!!!Congralution!");
    while(1)
    {

    }
  }

  if(numofplayer == 0&&Serial.available() > 0)
  {
    numofplayer = Serial.read();
    MsTimer2::set(30000, flash);        //30s
    MsTimer2::start();
    Serial.println(numofplayer - '0');
    Serial.println("The map: ");
    Serial.println("Y+++");
    Serial.println("++++");
    Serial.println("++++");
    Serial.println("++++");
    Serial.println("Your have 2 lifes now");
    Serial.println("Wait please");
  }

  if(numofplayer == 49)//single player
  {

   for (; numofmines<4; numofmines++)
       {
  x = rand()%4;
  y = rand()%4;

  if((x==0&&y==0) || (x==3&&y==3))
  numofmines--;
  else
  {
    if(mines[x][y]!=0)//have be set
    numofmines--;
    else
    {
      mines[x][y]=1;
    }
   }//set the mines
  }

  if (Serial.available() > 0)
  {
    incomingByte = Serial.read();
    if(incomingByte == 'w')//up
    {
      if(xnow == 0)
      {
        xnow=0;
      }
      else
      xnow--;

      Mines[xlast][ylast] = '+';//clear your last position
      Mines[xnow][ynow] = 'Y';//your present positon

      if(mines[xnow][ynow] == 1)
      yourlifes--;


      Serial.println(' ');
      for(int i=0;i<4;i++)
         for(int j=0;j<4;j++)
         {
          if(j!=3)
          Serial.print(Mines[i][j]);
          else
          Serial.println(Mines[i][j]);
         }

       if(yourlifes == 2)
       Serial.println("you have 2 lifes now");
       else if(yourlifes == 1)
       Serial.println("you have 1 lifes now");
       else
```

```
     Serial.println("you are over");

    incomingByte = 'x';
    xlast = xnow;
    ylast = ynow;
  }

  if(incomingByte == 's')//down
  {
    if(xnow == 3)
    {
      xnow=3;
    }
    else
    xnow++;

    Mines[xlast][ylast] = '+';//clear your last position
    Mines[xnow][ynow] = 'Y';//your present positon

    if(mines[xnow][ynow] == 1)
    yourlifes--;

   Serial.println(' ');
    for(int i=0;i<4;i++)
       for(int j=0;j<4;j++)
       {
        if(j!=3)
        Serial.print(Mines[i][j]);
        else
        Serial.println(Mines[i][j]);
       }

     if(yourlifes == 2)
     Serial.println("you have 2 lifes now");
     else if(yourlifes == 1)
     Serial.println("you have 1 lifes now");
     else
     Serial.println("you are over");

     //Serial.println(' ');
    incomingByte = 'x';
    xlast = xnow;
    ylast = ynow;
  }


  if(incomingByte == 'a')//left
  {
    if(ynow == 0)
    {
      ynow=0;
    }
    else
    ynow--;

    Mines[xlast][ylast] = '+';//clear your last position
    Mines[xnow][ynow] = 'Y';//your present positon
    if(mines[xnow][ynow] == 1)
    yourlifes--;


    Serial.println(' ');
    for(int i=0;i<4;i++)
       for(int j=0;j<4;j++)
       {
        if(j!=3)
        Serial.print(Mines[i][j]);
        else
        Serial.println(Mines[i][j]);
       }

     if(yourlifes == 2)
     Serial.println("you have 2 lifes now");
     else if(yourlifes == 1)
     Serial.println("you have 1 lifes now");
     else
     Serial.println("you are over");

    incomingByte = 'x';
    xlast = xnow;
    ylast = ynow;
  }


  if(incomingByte == 'd')//right
  {
    if(ynow == 3)
    {
      ynow=3;
    }
    else
    ynow++;

    Mines[xlast][ylast] = '+';//clear your last position
    Mines[xnow][ynow] = 'Y';//your present positon

    if(mines[xnow][ynow] == 1)
    yourlifes--;


    Serial.println(' ');
    for(int i=0;i<4;i++)
       for(int j=0;j<4;j++)
       {
        if(j!=3)
        Serial.print(Mines[i][j]);
        else
        Serial.println(Mines[i][j]);
       }

     if(yourlifes == 2)
     Serial.println("you have 2 lifes now");
     else if(yourlifes == 1)
     Serial.println("you have 1 lifes now");
     else
```

```
       Serial.println("you are over");

      incomingByte = 'x';
      xlast = xnow;
      ylast = ynow;
    }

  }



  }
  else if(numofplayer == 50)//two-player
  {
     if(firstTime == true && e!= 0)
  {
    Serial.print("Please enter four land mine's postion(1-14) for player1:\n");
  serialFlush();
  while(true){
    if(Serial.available()>0){
      tran_e = Serial.parseInt();
      Serial.println(tran_e);
      tran_b = Serial.parseInt();
      Serial.println(tran_b);
      tran_c = Serial.parseInt();
      Serial.println(tran_c);
      tran_d = Serial.parseInt();
      Serial.println(tran_d);
      break;
      }
      }
  firstTime = false;
  }

 if(firstTime == false)
 {
  while(true)
  {
    if(flag==1)//time is over
    {
      Serial.println("Time is over!Try again!");
      while(true)
      {

      }
    }

    if(yourlifes==0)
    {
      Serial.println("You are over!Try again!");
      while(true)
      {

      }
    }
    if(xnow==3&&ynow==3)
    {
      Serial.println("You win!Congralution!");
      while(true)
      {

      }
    }
  playGame();
  }
  }


  }

}
void flash()
{
  Serial.println("Time is over!!!");
  flag=1;
  MsTimer2::stop();
}

void playGame()
{
  //serialFlush();
  if (Serial.available() > 0)
  {
    incomingByte = Serial.read();
    if(incomingByte == 'w')//up
    {
      if(xnow == 0)
      {
        xnow=0;
      }
      else
      xnow--;

      Mines[xlast][ylast] = '+';//clear your last position
      Mines[xnow][ynow] = 'Y';//your present positon

      if(mines[xnow][ynow] == 1)
      yourlifes--;


      Serial.println(' ');
      for(int i=0;i<4;i++)
         for(int j=0;j<4;j++)
         {
          if(j!=3)
          Serial.print(Mines[i][j]);
          else
          Serial.println(Mines[i][j]);
         }

       if(yourlifes == 2)
       Serial.println("you have 2 lifes now");
       else if(yourlifes == 1)
       Serial.println("you have 1 lifes now");
       else
       Serial.println("you are over");
      incomingByte = 'x';
      xlast = xnow;
      ylast = ynow;
    }

    if(incomingByte == 's')//down
    {
      if(xnow == 3)
      {
        xnow=3;
      }
      else
      xnow++;

      Mines[xlast][ylast] = '+';//clear your last position
      Mines[xnow][ynow] = 'Y';//your present positon

      if(mines[xnow][ynow] == 1)
      yourlifes--;

     Serial.println(' ');
      for(int i=0;i<4;i++)
         for(int j=0;j<4;j++)
         {
          if(j!=3)
          Serial.print(Mines[i][j]);
          else
          Serial.println(Mines[i][j]);
         }

       if(yourlifes == 2)
       Serial.println("you have 2 lifes now");
       else if(yourlifes == 1)
       Serial.println("you have 1 lifes now");
       else
       Serial.println("you are over");

       //Serial.println(' ');
      incomingByte = 'x';
      xlast = xnow;
      ylast = ynow;
    }


    if(incomingByte == 'a')//left
    {
      if(ynow == 0)
      {
        ynow=0;
      }
      else
      ynow--;

      Mines[xlast][ylast] = '+';//clear your last position
      Mines[xnow][ynow] = 'Y';//your present positon

      if(mines[xnow][ynow] == 1)
      yourlifes--;


      Serial.println(' ');
      for(int i=0;i<4;i++)
         for(int j=0;j<4;j++)
         {
          if(j!=3)
          Serial.print(Mines[i][j]);
          else
          Serial.println(Mines[i][j]);
         }

       if(yourlifes == 2)
       Serial.println("you have 2 lifes now");
       else if(yourlifes == 1)
       Serial.println("you have 1 lifes now");
       else
       Serial.println("you are over");

      incomingByte = 'x';
      xlast = xnow;
      ylast = ynow;
    }


    if(incomingByte == 'd')//right
    {
      if(ynow == 3)
      {
        ynow=3;
      }
      else
      ynow++;

      Mines[xlast][ylast] = '+';//clear your last position
      Mines[xnow][ynow] = 'Y';//your present positon

      if(mines[xnow][ynow] == 1)
      yourlifes--;


      Serial.println(' ');
      for(int i=0;i<4;i++)
         for(int j=0;j<4;j++)
         {
          if(j!=3)
          Serial.print(Mines[i][j]);
          else
          Serial.println(Mines[i][j]);
         }

       if(yourlifes == 2)
       Serial.println("you have 2 lifes now");
       else if(yourlifes == 1)
       Serial.println("you have 1 lifes now");
       else
       Serial.println("you are over");
```

```
      incomingByte = 'x';
      xlast = xnow;
      ylast = ynow;
    }

  }
}

void receiveEvent(int howMany) {
  e=Wire.read();

  b=Wire.read();

  c=Wire.read();

  d=Wire.read();


  mines[e/4][e%4]=1;
  mines[b/4][b%4]=1;
  mines[c/4][c%4]=1;
  mines[d/4][d%4]=1;
}
void requestEvent()
{
 // Serial.println(tran_e);//test
 Wire.write(tran_e);
 Wire.write(tran_b);
 Wire.write(tran_c);
 Wire.write(tran_d);
}
void serialFlush(){
  while(Serial.available() > 0) {
    char t = Serial.read();
  }
}
```

```
//player2
#include <MsTimer2.h>
#include <Wire.h>

int hisxposition;
int hisyposition;
int myxposition;
int myyposition;
int hispunction;

int xnow = 0;
int ynow = 0;
int xlast = 0;
int ylast = 0;

int x;
int y;

int incomingByte = 0;
int numofplayer = 0;
int numofmines = 0;
int yourlifes = 2;
char Mines[4][4];
int mines[4][4] = {0};
int flag = 0;
int i=0;
int start=0;
int data[50];
int datain[50];
int symbol=0;
int symbol2=0;
int start2=0;

void flash()
{
  Serial.println("Time is over!!!");
  flag=1;
  MsTimer2::stop();
}

void serialFlush(){
  while(Serial.available() > 0) {
    char t = Serial.read();
  }
}

void playGame()
{

  if (Serial.available() > 0)
  {
    incomingByte = Serial.read();
    if(incomingByte == 'w')//up
    {
      if(xnow == 0)
      {
        xnow=0;
      }
      else
      xnow--;

      Mines[xlast][ylast] = '+';//clear your last position
      Mines[xnow][ynow] = 'Y';//your present positon

      if(mines[xnow][ynow] == 1)
      yourlifes--;


      Serial.println(' ');
      for(int i=0;i<4;i++)
         for(int j=0;j<4;j++)
         {
          if(j!=3)
          Serial.print(Mines[i][j]);
          else
          Serial.println(Mines[i][j]);
         }
```

```
     if(yourlifes == 2)
     Serial.println("you have 2 lifes now");
     else if(yourlifes == 1)
     Serial.println("you have 1 lifes now");
     else
     Serial.println("you are over");

    incomingByte = 'x';
    xlast = xnow;
    ylast = ynow;
  }

  if(incomingByte == 's')//down
  {
    if(xnow == 3)
    {
      xnow=3;
    }
    else
    xnow++;

    Mines[xlast][ylast] = '+';//clear your last position
    Mines[xnow][ynow] = 'Y';//your present positon

    if(mines[xnow][ynow] == 1)
    yourlifes--;

   Serial.println(' ');
    for(int i=0;i<4;i++)
       for(int j=0;j<4;j++)
       {
        if(j!=3)
        Serial.print(Mines[i][j]);
        else
        Serial.println(Mines[i][j]);
       }

     if(yourlifes == 2)
     Serial.println("you have 2 lifes now");
     else if(yourlifes == 1)
     Serial.println("you have 1 lifes now");
     else
     Serial.println("you are over");

     //Serial.println(' ');
    incomingByte = 'x';
    xlast = xnow;
    ylast = ynow;
  }


  if(incomingByte == 'a')//left
  {
    if(ynow == 0)
    {
      ynow=0;
    }
    else
    ynow--;

    Mines[xlast][ylast] = '+';//clear your last position
    Mines[xnow][ynow] = 'Y';//your present positon

    if(mines[xnow][ynow] == 1)
    yourlifes--;


    Serial.println(' ');
    for(int i=0;i<4;i++)
       for(int j=0;j<4;j++)
       {
        if(j!=3)
        Serial.print(Mines[i][j]);
        else
        Serial.println(Mines[i][j]);
       }

     if(yourlifes == 2)
     Serial.println("you have 2 lifes now");
     else if(yourlifes == 1)
     Serial.println("you have 1 lifes now");
     else
     Serial.println("you are over");

    incomingByte = 'x';
    xlast = xnow;
    ylast = ynow;
  }


  if(incomingByte == 'd')//right
  {
    if(ynow == 3)
    {
      ynow=3;
    }
    else
    ynow++;

    Mines[xlast][ylast] = '+';//clear your last position
    Mines[xnow][ynow] = 'Y';//your present positon

    if(mines[xnow][ynow] == 1)
    yourlifes--;


    Serial.println(' ');
    for(int i=0;i<4;i++)
       for(int j=0;j<4;j++)
       {
        if(j!=3)
        Serial.print(Mines[i][j]);
        else
        Serial.println(Mines[i][j]);
       }
```

```
        if(yourlifes == 2)
        Serial.println("you have 2 lifes now");
        else if(yourlifes == 1)
        Serial.println("you have 1 lifes now");
        else
        Serial.println("you are over");

       incomingByte = 'x';
       xlast = xnow;
       ylast = ynow;
     }

   }
 }

 void setup() {
   // put your setup code here, to run once:
   Serial.begin(9600);

   Serial.print("Please insert number of the players:");
    Wire.begin();
 /*
   Serial.println("The map: ");
   Serial.println("Y+++");
   Serial.println("++++");
   Serial.println("++++");
   Serial.println("++++");
   Serial.println('Your have 2 lifes now');
 */
   for(int i=0;i<4;i++)//initialize the array
      for(int j=0;j<4;j++)
      {
       if(i==0&&j==0)
       Mines[i][j] = 'Y';
       else
       {
         Mines[i][j] = '+';
       }

      }
   }

 void loop() {
   // put your main code here, to run repeatedly:
   if(flag==1)
   {
     while(1)
     {

     }
   }
   if(yourlifes == 0)
   {
     Serial.println(' ');
     Serial.println("You are over.Hhhhh");
     while(1)
     {
     }
   }

   if(xnow==3&&ynow==3)
   {
     Serial.println(' ');
     Serial.println("You win!!!Congralution!");
     while(1)
     {

     }
   }

   if(numofplayer == 0&&Serial.available() > 0)
   {
     numofplayer = Serial.read();
     MsTimer2::set(30000, flash);        // 30s
     MsTimer2::start();
     Serial.println(numofplayer - '0');
     Serial.println("The map: ");
     Serial.println("Y+++");
     Serial.println("++++");
     Serial.println("++++");
     Serial.println("++++");
     Serial.println("Your have 2 lifes now");

   }

   if(numofplayer == 49)//single player
   {

    for (; numofmines<4; numofmines++)
        {
   x = rand()%4;
   y = rand()%4;

   if((x==0&&y==0) || (x==3&&y==3))
   numofmines--;
   else
   {
     if(mines[x][y]!=0)//have be set
     numofmines--;
     else
     {
       mines[x][y]=1;
     }
    }//set the mines
   }

   if (Serial.available() > 0)
   {
     incomingByte = Serial.read();
     if(incomingByte == 'w')//up
     {
       if(xnow == 0)
       {
         xnow=0;
       }
       else
       xnow--;

       Mines[xlast][ylast] = '+';//clear your last position
       Mines[xnow][ynow] = 'Y';//your present positon

       if(mines[xnow][ynow] == 1)
       yourlifes--;


       Serial.println(' ');
       for(int i=0;i<4;i++)
          for(int j=0;j<4;j++)
          {
           if(j!=3)
           Serial.print(Mines[i][j]);
           else
           Serial.println(Mines[i][j]);
          }

        if(yourlifes == 2)
        Serial.println("you have 2 lifes now");
        else if(yourlifes == 1)
        Serial.println("you have 1 lifes now");
        else
        Serial.println("you are over");

       incomingByte = 'x';
       xlast = xnow;
       ylast = ynow;
     }

     if(incomingByte == 's')//down
     {
       if(xnow == 3)
       {
         xnow=3;
       }
       else
       xnow++;

       Mines[xlast][ylast] = '+';//clear your last position
       Mines[xnow][ynow] = 'Y';//your present positon

       if(mines[xnow][ynow] == 1)
       yourlifes--;

      Serial.println(' ');
       for(int i=0;i<4;i++)
          for(int j=0;j<4;j++)
          {
           if(j!=3)
           Serial.print(Mines[i][j]);
           else
           Serial.println(Mines[i][j]);
          }
        if(yourlifes == 2)
        Serial.println("you have 2 lifes now");
        else if(yourlifes == 1)
        Serial.println("you have 1 lifes now");
        else
        Serial.println("you are over");

        //Serial.println(' ');
       incomingByte = 'x';
       xlast = xnow;
       ylast = ynow;
     }


     if(incomingByte == 'a')//left
     {
       if(ynow == 0)
       {
         ynow=0;
       }
       else
       ynow--;

       Mines[xlast][ylast] = '+';//clear your last position
       Mines[xnow][ynow] = 'Y';//your present positon

       if(mines[xnow][ynow] == 1)
       yourlifes--;


       Serial.println(' ');
       for(int i=0;i<4;i++)
          for(int j=0;j<4;j++)
          {
           if(j!=3)
           Serial.print(Mines[i][j]);
           else
           Serial.println(Mines[i][j]);
          }

        if(yourlifes == 2)
        Serial.println("you have 2 lifes now");
        else if(yourlifes == 1)
        Serial.println("you have 1 lifes now");
        else
        Serial.println("you are over");

       incomingByte = 'x';
       xlast = xnow;
       ylast = ynow;
     }


     if(incomingByte == 'd')//right
     {
       if(ynow == 3)
       {
         ynow=3;
       }
       else
       ynow++;

       Mines[xlast][ylast] = '+';//clear your last position
       Mines[xnow][ynow] = 'Y';//your present positon

       if(mines[xnow][ynow] == 1)
       yourlifes--;


       Serial.println(' ');
       for(int i=0;i<4;i++)
          for(int j=0;j<4;j++)
          {
           if(j!=3)
           Serial.print(Mines[i][j]);
           else
           Serial.println(Mines[i][j]);
          }

        if(yourlifes == 2)
        Serial.println("you have 2 lifes now");
        else if(yourlifes == 1)
        Serial.println("you have 1 lifes now");
        else
        Serial.println("you are over");

       incomingByte = 'x';
       xlast = xnow;
       ylast = ynow;
     }

   }



   }






   else if(numofplayer==50)//two-player
   {

       if(Serial.available()>0)
       {
   int tran_e = Serial.parseInt();
   //Serial.println(tran_e);
   int tran_b = Serial.parseInt();
   //Serial.println(tran_b);
   int tran_c = Serial.parseInt();
   //Serial.println(tran_c);
   int tran_d = Serial.parseInt();
   //Serial.println(tran_d);
```

```
Wire.beginTransmission(4);
Wire.write(tran_e);
Wire.write(tran_b);
Wire.write(tran_c);
Wire.write(tran_d);
Wire.endTransmission();
Serial.print("wait for player2 setting up the initial mine field for you");
delay(8000); //waits for 8 seconds
Wire.requestFrom(4,8);

while(true){
if(Wire.available())
{
int e= Wire.read();
int b=Wire.read();
int c=Wire.read();
int d=Wire.read();

mines[e/4][e%4]=1;
mines[b/4][b%4]=1;
mines[c/4][c%4]=1;
mines[d/4][d%4]=1;

serialFlush();
while(true)
{
  if(flag==1)//time is over
  {
    Serial.println("Time is over!Try again!");
    while(true)
    {

    }
  }

  if(yourlifes==0)
  {
    Serial.println("You are over!Try again!");
    while(true)
    {

    }
  }
  if(xnow==3&&ynow==3)
  {
    Serial.println("You win!Congralution!");
    while(true)
    {

    }
  }
playGame();
}

}
```