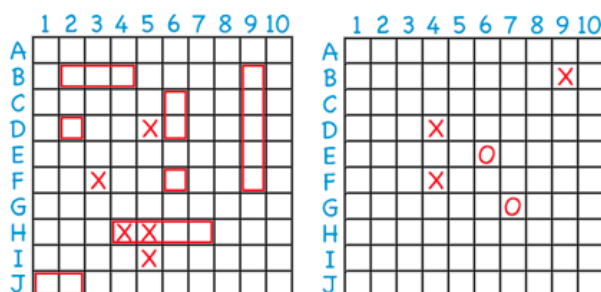


Matei-Sorin AXENTE (66967) - Battle Ships

Autorul poate fi contactat la adresa: **Login pentru adresa**

Introducere

Proiectul consta in crearea unei console pentru clasicul joc **Battle Ships** pentru doi jucatori, folosind doua matrici de LED-uri 8x8 RGB.



Ideea a venit de la un proiect din anii trecuti care implementa jocul Snake pe o matrice de LED-uri.

Utilitatea principala a proiectului este distractia si relaxarea printr-un joc clasic.

Descriere generală



Fiecare jucator va avea cate 5 butoane, 4 pentru deplasarea navelor in prima faza/mutarea cursorului pentru atac si unul pentru plasare nava/atac. Butoanele modifica starile matricilor de LED-uri prin deplasarea cursorului sau marcarea atacurilor, acestea alcatuind tabla de joc. Cardul SD va stoca efectele de sunet, ce vor insoti actiunile jucatorilor, in special atacurilor. LCD-ul text va tine scorul (nr casute ocupate de nave lovite de fiecare jucator) si va indruma jucatorii.

Functionalitate

La inceput, fiecare jucator isi va plasa setul de nave, de diferite dimensiuni, folosind 4 butoane pentru a se deplasa pe careul de 8x8 si un al cincilea buton pentru a confirma plasarea navei in acea pozitie.

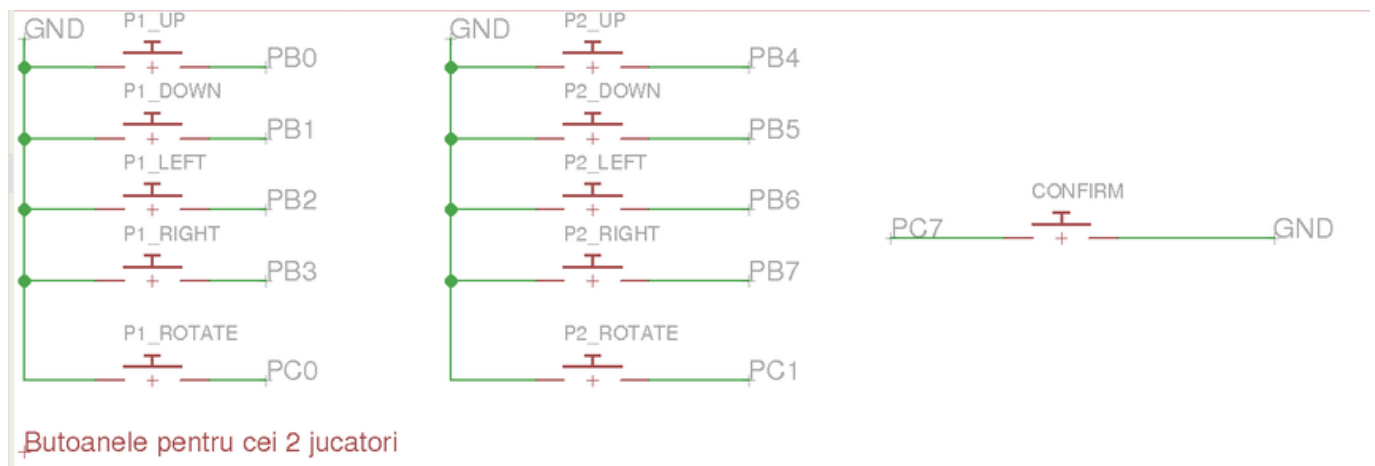
Cand ii vine randul, jucatorul activ va avea afisate atacurile precedente, pe matricea sa de LED-uri, alaturi de un cursor cu care va alege urmatorul atac. Jucatorul inactiv, va avea pe matricea de LED-uri

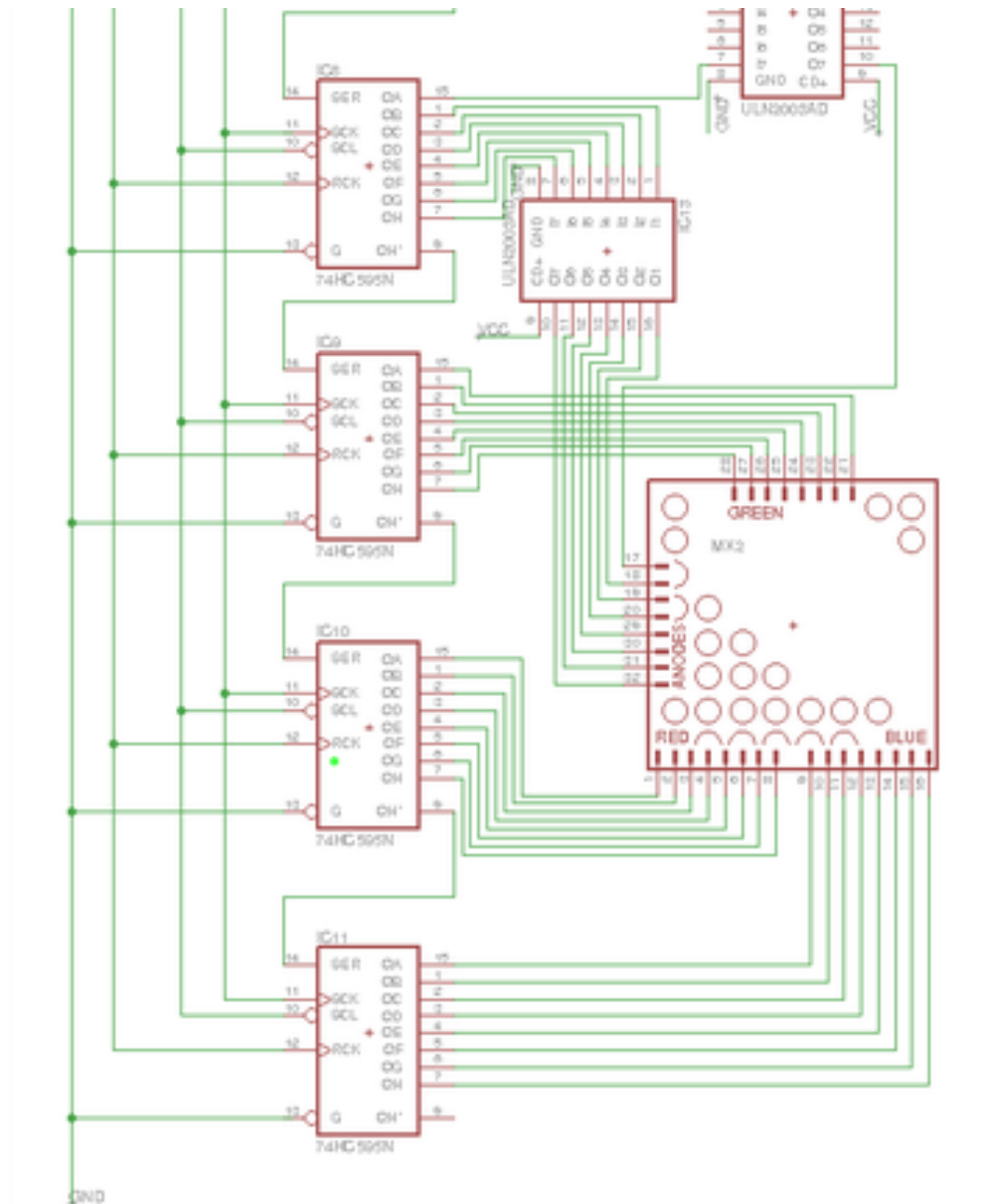
navele sale si atacurile adversarului.

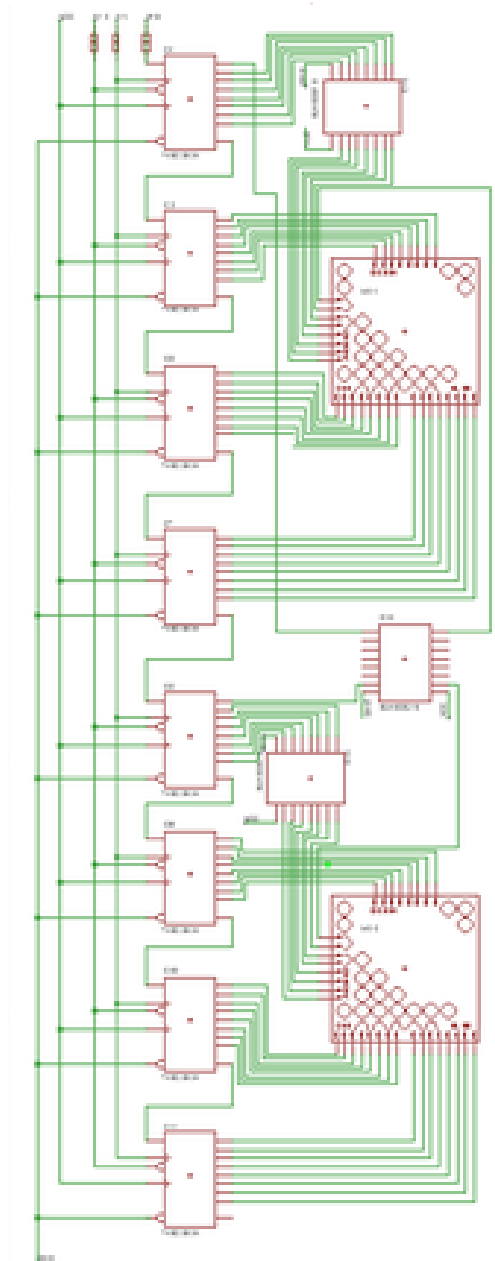
Jocul se termina cand un jucator reuseste sa distruga toate navele adversarului (loveste fiecare casuta ocupata de o nava).

Hardware Design

- Placa de baza x 1
- Matrice LED-uri 8×8 RGB x 2
- Butoane (diverse tipuri) x 10
- LCD text 16×2 x 1
- Shift Register 8 biti x 8
- Matrice tranzistoare x 3
- Buzzer x 1
- SD Card x 1







Software Design

Varianta Linux cu citire de la tastatura

```
#include <stdio.h>
#include <stdlib.h>

#define EMPTY '.'
#define TOBE 'o'
#define SHIP '0'

#define AIM '+'
#define HIT 'X'
#define MISS 'w'
```

```
void initMatrix(char ***m) {
    int p, i, j;

    *m = (char ***)malloc(2 * sizeof(char **));

    for(p = 0; p < 2; p++) {
        (*m)[p] = (char **)malloc(8 * sizeof(char *));
        for(i = 0; i < 8; i++)
            (*m)[p][i] = (char *)malloc(8 * sizeof(char));

        for(i = 0; i < 8; i++)
            for(j = 0; j < 8; j++)
                (*m)[p][i][j] = EMPTY;
    }
}

void clearMatrix(char ***m) {
    int p, i;
    for(p = 0; p < 2; p++) {
        for(i = 0; i < 8; i++)
            free((*m)[p][i]);
        free((*m)[p]);
    }
    free(*m);
}

void printMatrix(char **m) {
    int i, j;
    for(i = 0; i < 8; i++) {
        for(j = 0; j < 8; j++)
            printf("%c ", m[i][j]);
        printf("\n");
    }
}

void printBoth(char ***m) {
    int i, j;

    printf("\n");
    for(i = 0; i < 40; i++)
        printf("-");
    printf("\n");

    for(i = 0; i < 8; i++) {
        for(j = 0; j < 16; j++) {
            if(j == 8)
                printf("\t");

            if(j < 8)
                printf("%c ", m[0][i][j]);
        }
    }
}
```

```
        else
            printf("%c ", m[1][i][j - 8]);
    }
    printf("\n");
}

for(i = 0; i < 40; i++)
    printf("-");
printf("\n\n");
}

void switchPlayer(int *p) {
    *p = *p == 0 ? 1 : 0;
}

int placeShip(char ***m, int size, int *x, int *y, int dir) {
    int i = 0, rez = 1;
    if(dir == 1) {
        if(*x > 8 - size)
            *x = 8 - size;

        for(i = 0; i < size; i++)
            if((*m)[*x + i][*y] != SHIP)
                (*m)[*x + i][*y] = TOBE;
            else
                rez = 0;
    }
    else {
        if(*y > 8 - size)
            *y = 8 - size;

        for(i = 0; i < size; i++)
            if((*m)[*x][*y + i] != SHIP)
                (*m)[*x][*y + i] = TOBE;
            else
                rez = 0;
    }

    return rez;
}

void removeShip(char ***m) {
    int i, j;
    for(i = 0; i < 8; i++)
        for(j = 0; j < 8; j++)
            if((*m)[i][j] == TOBE)
                (*m)[i][j] = EMPTY;
}

void replace(char ***m) {
    int i, j;
```

```
        for(i = 0; i < 8; i++)
            for(j = 0; j < 8; j++)
                if((*m)[i][j] == TOBE)
                    (*m)[i][j] = SHIP;
    }

    int aim(char ***hits, char **opp, int x, int y) {
        if((*hits)[x][y] == EMPTY) {
            (*hits)[x][y] = AIM;
            return 1;
        }
        else
            return 0;
    }

    void check(char ***hits, char **opp, int x, int y, int *left) {
        if(opp[x][y] == SHIP) {
            (*hits)[x][y] = HIT;
            (*left)--;
        }
        else
            (*hits)[x][y] = MISS;
    }

    int main() {
        char action='\0', to;
        int i = 0, j = 0, change = 0, rez = 0;
    int rep = 0;

        int p = 0;

        char ***own;
        initMatrix(&own);

        char ***opp;
        initMatrix(&opp);

        int ships[6] = {4, 3, 3, 2, 2, 2};
        int placed[2] = {6, 6};
        int dir[2] = {0, 0};

        int left[2] = {16, 16};

        rez = placeShip(&own[p], ships[6 - placed[p]], &i, &j, dir[p]);

        while(placed[p] > 0) {
            printBoth(own);

            scanf("\n\n\n%c", &action);

            if(action == 'q')
```

```

        break;
    else if(action == 'c') {
        if(rez) {
            replace(&own[p]);
            placed[p]--;
            switchPlayer(&p);
            change = 1;
            i = 0;
            j = 0;
            rez = placeShip(&own[p], ships[6 -
placed[p]], &i, &j, dir[p]);
        }
        else
            printf("Invalid position\n");
    }
    else if(action == 'm') {
        if(left[p]) {
            removeShip(&own[p]);

            scanf(" %c", &to);

            switch(to) {
                case 's': i = (i < 8 ? i + 1 : 7);
break;

                case 'w': i = (i > 0 ? i - 1 : 0);
break;

                case 'd': j = (j < 8 ? j + 1 : 7);
break;

                case 'a': j = (j > 0 ? j - 1 : 0);
break;

            }

            rez = placeShip(&own[p], ships[6 -
placed[p]], &i, &j, dir[p]);
        }
        else if(action == 'r' && placed[p]) {
            removeShip(&own[p]);

            dir[p] = dir[p] == 0 ? 1 : 0;
            rez = placeShip(&own[p], ships[6 - placed[p]],
&i, &j, dir[p]);
        }
    }

    action = '\0';

    i = 0;
    j = 0;
    rez = aim(&opp[p], own[1 - p], i, j);

```

```

        while(left[p] > 0) {
            printf("SCORE: p1 - %d/16\tp2 - %d/16\n", left[0], left
[1]);

            printBoth(own);
            printBoth(opp);

            scanf("\n\n\n%c", &action);

            if(action == 'q')
                break;
            else if(action == 'c') {
                if(rez) {
                    check(&opp[p], own[1 - p], i, j, &left[
1 - p]);

                    switchPlayer(&p);
                    change = 1;
                    i = 0;
                    j = 0;
                    rez = aim(&opp[p], own[1 - p], i, j);
                }
                else
                    printf("Invalid position\n");
            }
            else if(action == 'm') {
                if(left[p]) {
                    if(opp[p][i][j] == AIM)
                        opp[p][i][j] = EMPTY;

                    scanf(" %c", &to);

                    switch(to) {
                        case 's': i = (i < 8 ? i + 1 : 7);

                        case 'w': i = (i > 0 ? i - 1 : 0);

                        case 'd': j = (j < 8 ? j + 1 : 7);

                        case 'a': j = (j > 0 ? j - 1 : 0);
                    }

                    rez = aim(&opp[p], own[1 - p], i, j);
                }
            }
        }

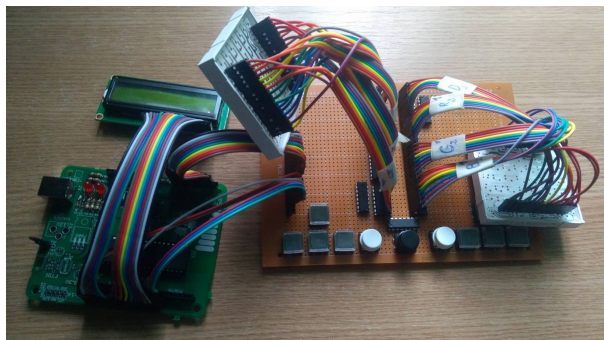
        printf("!!!!!!WINNER!!!!!!\n----PLAYER %d----\n", 1 - p);

        clearMatrix(&own);
        clearMatrix(&opp);

```

```
    return 0;  
}
```

Rezultate Obținute



Concluzii

Download

[335cc_axentemateisorin_final.sch](#)

[335cc_axentemateisorin.zip](#)

Jurnal

Bibliografie/Resurse

- Documentația în format [PDF](#)

Shift Registers [SN74HC595.pdf](#)

Tranzistor Matrix [ULN2003.html](#)

 8x8 RGB LED Matrix

From:

<http://ocw.cs.pub.ro/courses/> - **CS Open CourseWare**

Permanent link:

<http://ocw.cs.pub.ro/courses/pm/prj2017/astratulat/battleships>



Last update: **2021/04/14 15:07**